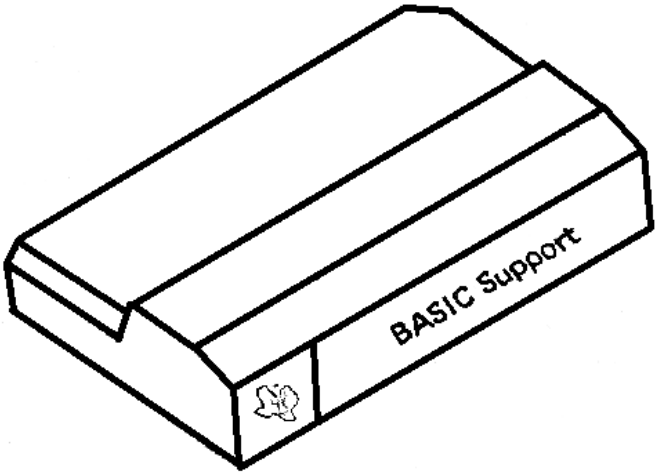
MORE POWER

from PCIF



TO YOUR 99/4

BLOCK MOVES, CHAINING,
CRU ACCESS, 2K BYTES CPU RAM
MACHINE CODE SUBROUTINES,
AND MUCH, MUCH, MORE

TI BASIC Support Module Pricing

with 2K RAM	- $150.00
without 2K RAM	- $100.00

Available to TI Cost Centers and individual TIer's and their families.

To order call Sally Hayes in Dallas, 995-7131.
Allow two weeks for delivery.

BASIC SUPPORT LIBRARY USER'S GUIDE
(PRELIMINARY)

INTRODUCTION

In order to form a more perfect union between the TI 99/4 and the BASIC programming language, this library provides access to various memory resources via subroutine calls from BASIC. These memory resources include VDP memory, CPU memory, CRU interface registers, VDP registers, and disk sectors.

The names of these support subroutines each begin with the character "@" in order to distinguish them as belonging to this collection.

Since two modules won't fit in one slot, the routines SAY and SPGET are also included for speech synthesizer interface.

SUMMARY

The BASIC support module was produced to enable an experienced programmer to get at the features of the 99/4 through BASIC. It does not require a memory extension.

Some of the major features are:

BLOCK MOVES: Allows for several screen images to be created and subsequently displayed rapidly. An example of use includes the agriculture demonstration where a screen full of data was quickly displayed rather than BASIC updating the screen line by line. Apparent screen motion can be obtained by rapidly updating the display memory.

EXECUTE 9900 CPU CODE: Allows a BASIC program to execute a very short (48 byte) 9900 program. This feature was used in generating a BASIC program to test the CPU memory extension.

PACKING UTILITIES: Allows for saving memory space. A 99/4 numeric variable takes 8 bytes. These routines allow 64 bits, 8 one-byte integers, or 4 two-byte integers to be stored into one location. This is very useful if the program requires a large number of small integers.

LOGICAL ROUTINES: BASIC does not allow logical operations, a frequent technique used by programmers. This will allow one to draw lines and curves and other graphics quickly from BASIC.

CRU ACCESS: Used in VTR to read tape position. Used in detecting telephone rings in TIVUE system. This is generally used for controlling devices and detecting external stimuli.

PROGRAM CHAINING: A facility is provided for saving a program on a peripheral device and subsequently re-starting the program when invoked from another program.

PLAY SOUND TABLE: The programmer may use these routines to construct and play sounds of more than the usual single chord, at maximum speed (60 sounds/second) and in parallel to other program activities.

BLOCK MOVE

Block move routines are supplied for moving large blocks of data within or between memory resources. The types of memory include VDP, RAM, CPU RAM or ROM, G-ROM.

The general form of a CALL to one of these routines is:

```
CALL @X2Y (Len, Src, Dest)
```

where X and Y are each V, C, or G. X represents the type of memory from which to move; Y represents the type into which to move.

V indicates VDP RAM
C indicates CPU (9900) memory address
G indicates a GROM

Len represents a variable whose value is the number of bytes to be transferred.

Src represents a variable whose value is the source (from) address.

Dest represents a variable whose value is the destination (to) address.

Note that Len, Src, and Dest are all variable names and may not be general expressions.

VDP ADDRESSING (V)

A variable containing a VDP address may have a value from 0 to 16383 (>3FFF). Moving to or from addresses outside this range can have undesired effects.

CPU ADDRESSING (C)

CPU (9900) memory has addresses from 0 to 65535 (>FFFF).

GROM ADDRESSING (G)

GROM addressing only applies to library module 0.
GROM has addresses from 0 to 65535 (>FFFF).
GROM as Dest is NOT ALLOWED.

Example 1:

To move a screen image into CPU memory at >2000,

```
L  = 768
V  = 0
C  = 8192
```

```
CALL @V2C (L,V,C)
```

Note: The screen image is at VDP RAM location 0 through 767 (>2FF).

Example 2:

To move the screen up 1 line (32 characters)

```
L  = 768 - 32
V1 = 32
V2 = 0
CALL @V2V (L,V1,V2)
```

BASIC VARIABLE ADDRESS

To acquire the VDP RAM address of a BASIC variable, use the @LOC routine.

```
CALL @LOC (Var,Addvar)
```

Var and Addvar represent two BASIC variable names. Th VDP RAM address of the first (low address) byte of Var is placed in Addvar.

Example: Save the screen image in array A.

```
DIM A (95)
CALL @LOC (A(o),AL)
L  = 768
V  = 0
CALL @V2V (L,V,AL)
```

NOTE: Each numeric variable requires 8 bytes so that 96 array elements (0 thru 95) are sufficient to hold 768 bytes.

NOTE: CALL @V2V (L,AL,V) will replace the saves screen image.

NOTE: CALL @LOC (A$,AL) will give the address of the string A$ in AL, but this value of AL must be used before the string A$ is re-allocated by the system. Numeric variables are not re-allocated!

CRU ACCESS

There are 2 routines for CRU access, one for reading, one for writing.

```
CALL @CRIN (Addr,Bits,Data)
CALL @CROUT (Addr,Bits,Data)
```

will cause the number of bits specified by Bits to be transferred from (to) the BASIC variable specified by Data into (from) the CRU at the address specified by Addr. The bits to be written should be right justified in DATA. The number of bits should be between 1 and 16.

EXECUTE 9900 (CPU) CODE:

```
CALL @CEX (R0,R1,R2)
```

Causes the 48 bytes at VDP RAM location >7D0 to execute as if it were a 9900 program at CPU RAM location >8300. This is strictly “user beware” and the 9900 program must have the following form:

- Locations >7D0, >7D1 must contain a pointer to the first instruction to be executed.

- If the workspace pointer is changed, then it must be restored to 83E0 before returning to BASIC.

- In order to return to BASIC, the last 9900 instruction executed must be:

```
B *R11     (>0458)
```

The register values at >8300, >8302, >8304 will be returned in R0,R1,R2 respectively.

The simplest way to get a program into VDP RAM location >7D0 is to use the CHAR function with character codes 154 thru 159.

<u>Example:</u>

```
CALL CHAR (154, “8302045B”)
CALL @CEX (A,B,C)
```

Will execute a return (B *R11) to BASIC, i.e., a NO-OP.

EXIT BASIC

To simulate a “SHIFT Q” from your program, execute the statement

```
CALL @EXIT
```

Nothing in memory will be recoverable.

MODIFY VDP REGISTER (Handle with Care)

To put the 1-byte value “Value” into the VDP register “Reg”,

```
CALL @VDRPUT (Reg,Value)
```

Where Reg and Value are BASIC variables containing the appropriate values.

ADJUST CHARACTER CODES

To modify the contents of a string of bytes in VDP RAM by adding a constant to each one,

```
CALL @VADD (L,Delta,Bytes)
```

LOGICAL OPERATIONS

```
CALL @VAND (L,V1,V2)
```

The string of bytes beginning at the address contained in V1 will be logically “AND” ed with (and the result stored into) the string of L bytes at the address contained in V2.

```
CALL @VOR (L,V1,V2)
```

Does the corresponding “OR” operation.

ONE-BIT FLAGS

To treat a numeric variable as 64 individual 1-bit flags;

```
CALL @ON (V,B)
```

To set bit B in variable V to one;

```
CALL @OFF (V,B)
```

To set bit B in variable V to zero;

```
CALL @TEST (V1,B,V2)
```

To get the value of bit B in variable V1 and put it in V2;

```
CALL @VON (V,B)
CALL @VOFF (V,B)
CALL @VTEST (V1,B,V2)
```

Function the same as @ON, @OFF, @TEST respectively, except that V is a numeric variable whose value is the address of the 8-byte VDP RAM area whose bit is to be set.

PACKING BYTES

If in any array of 8 numeric variables, each variable has an integer value less than 256, then these 8 values can be packed into a single numeric variable by

```
CALL @PUTB (A(X),B)
```

The 8 values in A(X),A(X+1) thru A(X+7) will be packed into the variable B, left to right (exponent first).

The reverse operation is accomplished by

```
CALL @GETB (B,A(X))
```

PACKING WORDS

Similar to Packing Bytes, except that 4 variables of integer value up to 2^{16-1} (65535) are packed into a single numeric variable.

Pack them in by

```
CALL @PUTW (A(X),B)
```

And unpack them by

```
CALL @GETW (B,A(X))
```

CHAINING

The BASIC SUPPORT MODULE provides a checkpoint and chaining facility. At any time after a BASIC program has been loaded into the TI 99/4, (whether it has been executed or not), the user may “checkpoint” that program onto an external device. This “checkpoint” file may be re-started from another BASIC program (chaining) at some later time. A re-started program will have retained all previously computed values of variables and all special character graphics for character codes above 95.

CHECKPOINT

```
CALL CHECK (VAR$)
```

VAR$ is the name of a string variable whose value is the name of the file onto which to save the current program and associated parameters.

NOTE: The call to CHECK must be an imperative statement, not a program statement.

RESTART

```
CALL CHAIN (VAR$)
```

VAR$ is the name of a string variable whose value is the name of the file from which to retrieve a program to be re-started.

NOTE: The call to CHAIN must be a program statement, not an imperative statement.

PLAY SOUND TABLE

```
CALL @PLAY (VADD)
```

VADD is the name of a numeric variable whose value is the VDP RAM address at which is located the sound table to be “played”. This “playing” will continue until a “stop” code is encountered or until another CALL @PLAY is executed, or CALL SOUND, or any system function which causes a “beep”.

```
CALL @PLABK (VADD)
```

If this statement is executed immediately after a CALL SOUND statement, an 11-byte string of sound definition and a 6-byte “stop”, string will be placed at the VDP RAM location whose address is the value of VADD.

ACQUIRE SYSTEM DEPENDENT VALUES

The following subroutines may be called to acquire the corresponding values:

@MEM	CURRENT AVAILABLE VDP RAM
@MAXI	MAXIMUM POSITIVE VALUE
@MINI	LARGEST NEGATIVE VALUE
@EPS	SMALLEST POSITIVE NUMBER
@BIGI	LARGEST CONSECUTIVE INTEGER

Example:

```
CALL @MEM (A)
```

will cause the value of A to be changed to the number of bytes remaining available to strings, subroutine stack, etc., at the time of the CALL.

DISPLAY A LINE OF TEXT

CALL @AT (R,C,T$) will display the string value of the variable T$ horizontally from left to right on the display screen (with “LF/CR” after column 32) beginning at the row and column indicated by the values of variables R and C respectively.

Specifically, @AT acts exactly like HCHAR, with T$ substituting for the string of multiple copies of the same character.

CALL A KNOWN GPL SUBROUTINE

CALL @GSUB (GADDR) will transfer to the GPL subroutine located at the address indicated by the value of the variable GADDR. Some examples of useful subroutines are:

ADDRESS	ROUTINE
>201C	REDEFINE DEFAULT CHARACTER SET
>9D18	STRING-SPACE GARBAGE COLLECTION

```
                 1            TITLE 'BSM'
                 2   *    BASIC SUPPORT MODULE
                 3   *    PHASE 1 FINAL VERSION
                 4   *        11/26/80
                 5   *
                41            GROM    3
                42            ORG     0
6000 AA0101     43            DATA    >AA,1,1,0,#0,#PROG1,#0,#SUBS,#0,#0
6003 000000
6006 667000
6009 006010
600C 000000
600F 00
     6010       44   SUBS     EQU     $
6010 602660     45   V2V      DATA    #CRIN,V2V01
6013 19
6014 044056     46            DATA    4,:@V2V:
6017 3256
6019 066125     47   V2V01    CALL    P
601C 340280     48            MOVE    @Q1 FROM RAM(@Q2) TO RAM(@Q3)
601F 0AB006
6022 060012     49            CALL    RPL
                50   ************************************************************
6025 607560     51   CRIN     DATA    #CROUT,#CRIN01
6028 3B
6029 054043     52            DATA    5, :@CRIN:
                53   *
                54   *        MACHINE LANGUAGE (CRINDATA):
                55   * >830C      MOV   @>8302,R12      CRU ADDRESS
                56   * >8310      SLA   R12,1           TIMES 2
                57   * >8312      STCR  @>8300,0        READ 16 BITS INTO >8300
                58   * >8316      B     *R11            RETURN TO G-LANGUAGE
                59   *
602F C32083     60   CRINDATA  DATA    #>C320,#>8302,#>0A1C,#>3420,#>8300,#>045B
6032 020A1C
6035 342083
6038 00045B
                61            ENTRY POINT FROM BASIC
603B 066125     62   CRIN01   CALL    P
603E BD0A00     63            DST     @0,@Q3          SAVE RTN BUFFER POINTER
6041 BD0806     64            DST     @Q2,@Q2+2
6044 06605D     65            CALL    CRIN02
6047 CF0800     66            $IF     @Q2+2 .DGT. #>8 GOTO CRIN03
604A 086051
604D E70000     67            DSRL    @0,8
6050 08
6051 0665A1     68   CRIN03   CALL    CIF             CONV INTEGER TO FLOATING
6054 350008     69            MOVE    8 FROM @FAC TO RAM(@Q3)
6057 B00A4A
605A 060012     70            CALL    RPL
                71   *
                72   *        ENTRY POINT FROM G-LANGUAGE
                73   *        INPUT:
                74   *              CRU ADDR IS IN      Q1,Q1+1
```

```
                 *              # BITS   IS IN     @2+1
                 *        OUTPUT:
                 *              CRU PATTERN        @>00
                 *
605D 31000C      CRIN02   MOVE   12 FROM ROM(#CRINDATA) TO @>C
6060 0C602F
6063 BF0083      DST     >830C,@0        POINTER TO MACHINE LANG CODE
6066 0C
6067 B30600      DAND    >000F,@02       MARK OFF 12 BITS
606A 0F
606B E30600      DSLL    @02,6           SHIFT # BITS INTO POSITION
606E 06
606F B51206      DOR     @02,@>12        COMBINE STCR INSTR & # BITS
6072 0FF0        XML     >F0
6074 00          RTN
                 ***************************************************************
6075 609960      CROUT    DATA   #CEX,#CROT01
6078 80
6079 064043      DATA    6,:@CROUT:
607C 624F55
607F 54
                 *        ENTRY POINT FROM BASIC
6080 066125      CROT01   CALL   P
6083 066089      CALL    CROT02
6086 060012      CALL    RPL
                 *
                 *        ENTRY POINT FROM G-LANGUAGE
                 *        INPUT:
                 *               CRU ADDR IS IN      @1,@1+1
                 *               # BITS   IS IN     @2+1
                 *               CRU PATTERN IS IN   @3,@3+1
                 *        OUTPUT:
                 *               NONE
                 *
6089 BD6802      CROT02   DST     @01,@IOB
608C BC6A07      ST      @02+1,@IOB+2
608F BE686C      ST      BUF,@IOB+3
6092 BD6C0A      DST     @03,@BUF
6095 F66803      I/O     @IOB,3
6098 00          RTN
                 ***************************************************************
6099 60A260      CEX      DATA   #CEXP,@CEX01
609C DF
609D 044043      DATA    4,:@CEX:
60A0 4558
60A2 618260      CEXP     DATA   #LOC,#CEXP01
60A5 AC
60A6 054043      DATA    5,:@CEXP:
60A9 455850
60AC 066125      CEXP01   CALL   P
6CAF BD0002      DST     @01,@>0
60B2 0FF0        XML     >F0
60B4 8F0060      $IF     @>0 .DEQ. 0 GOTO CEXP02
60B7 DC
60B8 BD0006      DST     @02,@>0
```

```
60BB 0665A1  118            CALL    CIF
60BE 350008  119            MOVE    8 FROM @FAC TO RAM(@Q2+2)
60C1 B008AA
60C4 BD000A  120            DST     @Q3,@>0
60C7 0665A1  121            CALL    CIF
60CA 350008  122            MOVE    8 FROM @FAC TO RAM(@Q3+2)
60CD B00C4A
60D0 BD000E  123            DST     @Q4,@>0
60D3 0665A1  124            CALL    CIF
60D6 350008  125            MOVE    8 FROM @FAC TO RAM(@Q4+2)
60D9 B0104A
60DC 060012  126  CEXP02    CALL    RPL
60DF 066625  127  CEX01     CALL    PGMCTR
60E2 35001A  128            MOVE    26 FROM @18 TO RAM(ROLOUT)
60E5 A3C016
60E8 350030  129            MOVE    48 FROM RAM(>7D0) TO @0
60EB 00A7D0
60EE 0FF0    130            XML     >F0
60F0 35001A  131            MOVE    26 FROM RAM(ROLOUT) TO @18
60F3 18A3C0
60F6 3E0006  132            MOVE    6 FROM @0 TO @10
60F9 1000
60FB 066128  133            CALL    PA
60FE BD0010  134            DST     @>10,@0
6101 0665A1  135            CALL    CIF
6104 350008  136            MOVE    8 FROM @FAC TO RAM(@Q1+2)
6107 B0044A
610A BD0012  137            DST     @>12,@0
610D 0665A1  138            CALL    CIF
6110 350008  139            MOVE    8 FROM @FAC TO RAM(@Q2+2)
6113 B0084A
6116 BD0014  140            DST     @>14,@0
6119 0665A1  141            CALL    CIF
611C 350008  142            MOVE    8 FROM @FAC TO RAM(@Q3+2)
611F B00C4A
6122 060012  143            CALL    RPL
     6125    144  P         EQU     $
6125 066625  145            CALL    PGMCTR
6128 D642B7  146  PA        $IF     @CHAT .NE. LPAR$ GOTO ERR7
612B 460D
612D 0F1B    147            XML     PGMCHR
612F 066171  148            CALL    PARM
6132 BD024A  149            DST     @FAC,@Q1
6135 BD0400  150            DST     @0,@Q1+2
6138 D642B6  151            $IF     @CHAT .EQ. RPAR$ GOTO FINIS
613B 6167
613D 06616A  152            CALL    COMMA2
6140 BD064A  153            DST     @FAC,@Q2
6143 BD0800  154            DST     @0,@Q2+2
6146 D642B6  155            $IF     @CHAT .EQ. RPAR$ GOTO FINIS
6149 6167
614B 06616A  156            CALL    COMMA2
614E BD0A4A  157            DST     @FAC,@Q3
6151 BD0C00  158            DST     @0,@Q3+2
6154 D642B5  159            $IF     @CHAT .EQ. RPAR$ GOTO FINIS
```

```
6157 6167
6159 066116A  160            CALL    COMMA2
615C BD0E4A   161            DST     @FAC,@Q4
615F BD1000   162            DST     @0,@Q4+2
6162 D642B6   163            $IF     @CHAT .NE. RPAR$ GOTO ERR7
6165 460D
6167 0F1B     164  FINIS     XML     PGMCHR
6169 00       165            RTN
              166  *
              167  *         PARAMETER FETCH
              168  *
616A D642B3   169  COMMA2    $IF     @CHAT .NE. COMMA$ GOTO ERR7
616D 460D
616F 0F1B     170            XML     PGMCHR
6171 0F13     171  PARM      XML     SYM
6173 0F14     172            XML     SMB
6175 BD004E   173            DST     @FAC+4,@0
6178 350008   174            MOVE    8 FROM RAM(@0) TO @FAC
617B 4AB000
617E 0665CF   175            CALL    CVFLIN
6181 00       176            RTN
6182 619D61   177  LOC       DATA    #G2V,#LOC01
6185 8B
6186 044044   178            DATA    4,:@LOC:
6189 4F43
618B 066125   179  LOC01     CALL    P
618E BD0004   180            DST     @Q1+2,@0
6191 0665A1   181            CALL    CIF
6194 350008   182            MOVE    8 FROM @FAC TO RAM(@Q2+2)
6197 B0084A
619A 060012   183            CALL    RPL
619D 61B361   184  G2V       DATA    #C2V,#G2V01
61A0 A6
61A1 044047   185            DATA    4,:@G2V:
61A4 3256
61A6 066125   186  G2V01     CALL    P
61A9 3202B0   187            MOVE    @Q1 FROM ROM(@Q2) TO RAM(@Q3)
61AC 0A0000
61AF 06
61B0 060012   188
61B3 61CD61   189  C2V       DATA    #V2G,#C2V01
61B6 BC
61B7 044043   190            DATA    4,:@C2V:
61BA 3256
61BC 066125   191  C2V01     CALL    P
61BF A70683   192            DSUB    CBIAS,@Q2
61C2 00
61C3 3402B0   193            MOVE    @Q1 FROM @0(Q2) TO RAM(@Q3)
61C6 0AC000
61C9 06
61CA 060012   194            CALL    RPL
61CD 61E361   195  V2G       DATA    #V2C,#V2G01
61D0 D6
61D1 044056   196            DATA    4,:@V2G:
61D4 3247
```

```
61D6 066125  197  V2G01    CALL   P
61D9 2C0200  198           MOVE   @Q1 FROM RAM(@Q2) TO ROM(@Q3)
61DC 000AB0
61DF 06
61E0 060012  199           CALL   RPL
61E3 61FD61  200  V2C      DATA   #C2C,#V2C01
61E6 EC
61E7 044056  201           DATA   4,:@V2C:
61EA 3243
61EC 066125  202  V2C01    CALL   P
61EF A70A83  203           DSUB   CBIAS,@Q3
61F2 00
61F3 3402C0  204           MOVE   @Q1 FROM RAM(@Q2) TO @0(Q3)
61F6 000AB0
61F9 06
61FA 060012  205           CALL   RPL
61FD 621C62  206  C2C      DATA   #G2G,#C2C01
6200 06
6201 044043  207           DATA   4,:@C2C:
6204 3243
6206 066125  208  C2C01    CALL   P
6209 A70683  209           DSUB   CBIAS,@Q2
620C 00
620D A70A83  210           DSUB   CBIAS,@Q3
6210 00
6211 3402C0  211           MOVE   @Q1 FROM @0(Q2) TO @0(Q3)
6214 000AC0
6217 0006
6219 060012  212           CALL   RPL
621C 623362  213  G2G      DATA   #C2G,#G2G01
621F 25
6220 044047  214           DATA   4,:@G2G:
6223 3247
6225 066125  215  G2G01    CALL   P
6228 2A0200  216           MOVE   @Q1 FROM ROM(@Q2) TO ROM(@Q3)
622B 000A00
622E 0006
6230 060012  217           CALL   RPL
6233 624E62  218  C2G      DATA   #G2C,#C2G01
6236 3C
6237 044043  219           DATA   4,:@C2G:
623A 3247
623C 066125  220  C2G01    CALL   P
623F A70683  221           DSUB   CBIAS,@Q2
6242 00
6243 2C0200  222           MOVE   @Q1 FROM @0(Q2) TO ROM(@Q3)
6246 000AC0
6249 0006
624B 060012  223           CALL   RPL
624E 626962  224  G2C      DATA   #CRUN,#G2C01
6251 57
6252 044047  225           DATA   4,:@G2C:
6255 3243
6257 066125  226  G2C01    CALL   P
625A A70A83  227           DSUB   CBIAS,@Q3
```

```
625D 00
625E 3202C0  228          MOVE   @01 FROM ROM(@02) TO @0(03)
6261 000A00
6264 0006
6266 060012  229          CALL   RPL
6269 627B62  230  CRUN    DATA   #VDRPUT,#CRUN01
626C 73
626D 054043  231          DATA   5,:@CRUN:
6270 52554E
6273 066625  232  CRUN01  CALL   PGMCTR
6276 0F20    233          XML    C2K
6278 060012  234          CALL   RPL
627B 62B562  235  VDRPUT  DATA   #VADD,#VDRP01
627E 99
627F 074056  236          DATA   7,:@VDRPUT:
6282 445250
6285 5554
6287 C06083  237  VDRDATA DATA   #>C060,#>8302                    MOV >8302,R
628A 02
628B 06C1D8  238          DATA   #>06C1,#>D801,#>8C02             SWPB: MOVB
628E 018C02
6291 06C1D8  239          DATA   #>06C1,#>D801,#>8C02,#>045B      SWPB: MOVB
6294 018C02
6297 045B
6299 066125  240  VDRP01  CALL   P
629C E30200  241          DSLL   @01,8
629F 08
62A0 BC0307  242          ST     @02+1,@01+1
62A3 B60280  243          SB     @01,7
62A6 310012  244          MOVE   18 FROM ROM(#VDRDATA) TO @>4
62A9 046287
62AC BF0083  245          DST    >8304,@0
62AF 04
62B0 0FF0    246          XML    >F0
62B2 060012  247          CALL   RPL
             248  *
62B5 62D662  249  VADD    DATA   #VOR,#VADD01
62B8 BF
62B9 054056  250          DATA   5,:@VADD:
62BC 414444
62BF 066125  251  VADD01  CALL   P
62C2 A1020A  252          DADD   @03,@01
62C5 D10A02  253          $WHILE @03 .DLT. @01                          1
62C8 62D3
62CA A0B00A  254          ADD    @02+1,RAM(@03)                         1
62CD 07
62CE 910A    255          DINC   @03                                    1
62D0 0562C5  256          $END
62D3 060012  257          CALL   RPL
62D6 62F962  258  VOR     DATA   #VAND,#VOR01
62D9 DF
62DA 044056  259          DATA   4,:@VOR:
62DD 4F52
62DF 066125  260  VOR01   CALL   P
62E2 A1020A  261          DADD   @03,@01
```

```
62E5 D10A02  262          $WHILE  @Q3 .DLT. @Q1                          1
62E8 62F6
62EA B4B00A  263          OR      RAM(@Q2),RAM(@Q3)                      1
62ED B006
62EF 910A    264          DINC    @Q3                                    1
62F1 9106    265          DINC    @Q2                                    1
62F3 0562E5  266          $END
62F6 060012  267          CALL    RPL
62F9 631D63  268  VAND    DATA    #RSECT,#VAND01
62FC 03
62FD 054056  269          DATA    5,:@VAND:
6300 414E44
6303 066125  270  VAND01  CALL    P
6306 A1020A  271          DADD    @Q3,@Q1
6309 010A02  272          $WHILE  @Q3 .DLT. @Q1                          1
630C 631A
630E B0B00A  273          AND     RAM(@Q2),RAM(@Q3)                      1
6311 B006
6313 910A    274          DINC    @Q3                                    1
6315 9106    275          DINC    @Q2                                    1
6317 056309  276          $END
631A 060012  277          CALL    RPL
             278  *
     0030    279  STLN    EQU     >30
     0032    280  ENLN    EQU     >32
     0000    281  SRC     EQU     0
     0002    282  DEST    EQU     2
     0070    283  PTOP    EQU     >70
     2840    284  PRGLST  EQU     >2840
     03EC    285  SAVEPC  EQU     >3EC
     0024    286  STVSPT  EQU     >24
     05F8    287  VRAMVS  EQU     >5F8
     0044    288  BUFFY   EQU     >44
     0034    289  DATA    EQU     >34
     006E    290  VSPTR   EQU     >6E
     0006    291  PLEN    EQU     Q2
     0008    292  BLEN    EQU     PLEN+2
             293  *
             294  *** CALL @RSEC(UNIT,SECTOR,BUFFER,ERROR)
             295  *
631D 635763  296  RSECT   DATA    #WSECT,#RSEC01
6320 27
6321 054052  297          DATA    5,:@RSEC:
6324 534543
6327 066125  298  RSEC01  CALL    P
632A BF4C00  299          DST     >00FF,@FAC+2
632D FF
632E BF4A00  300  RSEC02  DST     >0000,@FAC
6331 00
6332 BC4C03  301          ST      @Q1+1,@FAC+2
6335 BD4E0A  302          DST     @Q3,@FAC+4
6338 BD5006  303          DST     @Q2,@FAC+6
633B BFA000  304          DST     >0110,RAM(0)
633E 0110
6340 BF5600  305  RSEC03  DST     >0000,@>56
```

```
6343 00
6344 060010  306          CALL   >10
6347 0A      307          DATA   >0A
6348 BD0050  308          DST    @FAC+6,@0
634B 0665A1  309          CALL   CIF
634E 350008  310          MOVE   8 FROM @FAC TO RAM(@Q4+2)
6351 B0104A
6354 060012  311          CALL   RPL
             312  *
             313  *** CALL @WSEC(UNIT,SECTOR,BUFFER,ERROR)
             314  *
6357 636B63  315  WSECT   DATA   #RENAME,#WSEC01
635A 61
635B 054057  316          DATA   5,:@WSEC:
635E 534543
6361 066125  317  WSEC01  CALL   P
6364 BF4C00  318          DST    >0000,@FAC+2
6367 00
6368 05632E  319          B      RSEC02
             320  *
             321  *** CALL @RENAME(UNIT,OLDNAME,NEWNAME,ERROR)
             322  *
636B 638B63  323  RENAME  DATA   #DSKFMT,#RENM01
636E 77
636F 074052  324          DATA   7,:@RENAME:
6372 454E41
6375 4D45
6377 066125  325  RENM01  CALL   P
637A BC4C03  326          ST     @Q1+1,@FAC+2          UNIT NUMBER
637D BD5008  327          DST    @Q2+2,@FAC+6          OLD NAME
6380 BD4E0C  328          DST    @Q3+2,@FAC+4          NEW NAME
6383 BFA000  329          DST    >0113,RAM(0)
6386 0113
6388 056340  330          B      RSEC03
             331  *
             332  ***   CALL @DSKFMT(UNIT,TRACKS,BUFFER,ERROR)
             333  *
638B 63AF63  334  DSKFMT  DATA   #ON,#DFMT01
638E 97
638F 074044  335          DATA   7,:@DSKFMT:
6392 534B46
6395 4D54
6397 066125  336  DFMT01  CALL   P
639A BF4A00  337          DST    >0000,@FAC
639D 00
639E BD4C05  338          DST    @Q2,@FAC+2            NUMBER OF TRACKS
63A1 BD4C03  339          ST     @Q1+1,@FAC+2          UNIT NUMBER
63A4 BD4E0A  340          DST    @Q3,@FAC+4            BUFFER AREA
63A7 BFA000  341          DST    >0111,RAM(0)
63AA 0111
63AC 056340  342          B      RSEC03
             343  *
63AF 63B340  344  ON      DATA   #VON,#ON01        SET BIT "ON"
63B2 C9
63B3 03404F  345          DATA   3,:@ON:
```

```
63B6 4E
63B7 63DC63  346  VON      DATA   #OFF,#VON01      SET BIT "ON" INDIRECT
63BA C0
63BB 044056  347           DATA   4,:@VON:
63BE 4F4E
63C0 066125  348  VON01    CALL   P
63C3 BD0402  349           DST    @Q1,@Q1+2
63C6 0563CC  350           B      ON02
63C9 066125  351  ON01     CALL   P
63CC 06650C  352  ON02     CALL   BYTBIT
63CF BE0280  353           ST     >80,@Q1
63D2 E80207  354           SRC    @Q1,@Q2+1
63D5 B4B004  355           OR     @Q1,RAM(@Q1+2)
63D8 02
63D9 060012  356           CALL   RPL
             357  *
63DC 63E563  358  OFF      DATA   #VOFF,#OFF01
63DF F8
63E0 04404F  359           DATA   4,:@OFF:
63E3 4646
63E5 640B63  360  VOFF     DATA   #TEST,#VOFF01
63E8 EF
63E9 054056  361           DATA   5,:@VOFF:
63EC 4F4646
63EF 066125  362  VOFF01   CALL   P
63F2 BD0402  363           DST    @Q1,@Q1+2
63F5 0563FB  364           B      OFF02
63F8 066125  365  OFF01    CALL   P
63FB 06650C  366  OFF02    CALL   BYTBIT
63FE BE027F  367           ST     >7F,@Q1
6401 E80207  368           SRC    @Q1,@Q2+1
6404 B0B004  369           AND    @Q1,RAM(@Q1+2)
6407 02
6408 060012  370           CALL   RPL
             371  *
640B 641564  372  TEST     DATA   #VTEST,#TEST01
640E 26
640F 054054  373           DATA   5,:@TEST:
6412 455354
6415 644C64  374  VTEST    DATA   #GETWRD,#VTST01
6418 20
6419 064056  375           DATA   6,:@VTEST:
641C 544553
641F 54
6420 BD0402  376  VTST01   DST    @Q1,@Q1+2
6423 056429  377           B      TEST02
6426 066125  378  TEST01   CALL   P
6429 06650C  379  TEST02   CALL   BYTBIT
642C BE0280  380           ST     >80,@Q1
642F E80207  381           SRC    @Q1,@Q2+1
6432 860E    382           CLR    @Q4
6434 350007  383           MOVE   7 FROM @Q4 TO @Q4+1
6437 0F0E
6439 D802B0  384           CLOG   RAM(@Q1+2),@Q1
643C 04
```

```
643D 6443    385            BS      UNSET
643F BF0EBF  386            DST     >BFFF,@Q4
6442 FF
6443 350008  387  UNSET     MOVE    8 FROM @Q4 TO RAM(@Q3+2)
6446 B00C0E
6449 060012  388            CALL    RPL
644C 647C64  389  GETWRD    DATA    #PUTWRD,#GETW01
644F 56
6450 054047  390            DATA    5,:@GETW:
6453 455457
6456 066125  391  GETW01    CALL    P
6459 870A    392            DCLR    @Q3
645B CB0A00  393            $WHILE  @Q3 .DL. >4                                  1
645E 046479
6461 BD00B0  394            DST     RAM(@Q1+2),@0                                1
6464 04
6465 0665A1  395            CALL    CIF                                          1
6468 350008  396            MOVE    8 FROM @FAC TO RAM(@Q2+2)                    1
646B B0084A
646E 9504    397            DINCT   @Q1+2                                        1
6470 A30800  398            DADD    8,@Q2+2                                      1
6473 08
6474 910A    399            DINC    @Q3                                          1
6476 05645B  400            $END
6479 060012  401            CALL    RPL
647C 64AB64  402  PUTWRD    DATA    #GETBYT,#PUTW01
647F 86
6480 054050  403            DATA    5,:@PUTW:
6483 555457
6486 066125  404  PUTW01    CALL    P
6489 870A    405            DCLR    @Q3
648B CB0A00  406            $WHILE  @Q3 .DL. >4                                  1
648E 0464A8
6491 350008  407            MOVE    8 FROM RAM(@Q1+2) TO @FAC                    1
6494 4AB004
6497 0F12    408            XML     CFI                                          1
6499 BDB008  409            DST     @FAC,RAM(@Q2+2)                              1
649C 4A
649D 9508    410            DINCT   @Q2+2                                        1
649F A30400  411            DADD    8,@Q1+2                                      1
64A2 08
64A3 910A    412            DINC    @Q3                                          1
64A5 05648B  413            $END
64A8 060012  414            CALL    RPL
64AB 64DD64  415  GETBYT    DATA    #PUTBYT,#GETB01
64AE B5
64AF 054047  416            DATA    5,:@GETB:
64B2 455442
64B5 066125  417  GETB01    CALL    P
64B8 870A    418            DCLR    @Q3
64BA CB0A00  419            $WHILE  @Q3 .DL. >8                                  1
64BD 0864DA
64C0 8700    420            DCLR    @0                                           1
64C2 BC0180  421            ST      RAM(@Q1+2),q1                                1
64C5 04
```

```
64C6 0665A1  422            CALL    CIF                                    1
64C9 350008  423            MOVE    8 FROM @FAC TO RAM(@Q2+2)              1
64CC B0084A
64CF 9104    424            DINC    @Q1+2                                  1
64D1 A30800  425            DADD    8,@Q2+2                                1
64D4 08
64D5 910A    426            DINC    @Q3                                    1
64D7 0564BA  427            $END
64DA 060012  428            CALL    RPL
64DD 651A64  429  PUTBYT    DATA    #RUN,#PUTB01
64E0 E7
64E1 054050  430            DATA    5,:@PUTB:
64E4 555442
64E7 066125  431  PUTB01    CALL    P
64EA 870A    432            DCLR    @Q3
64EC CB0A00  433            $WHILE  @Q3 .DL. >8                            1
64EF 086509
64F2 350008  434            MOVE    8 FROM RAM(@Q1+2) TO @FAC              1
64F5 4AB004
64F8 0F12    435            XML     CFI                                    1
64FA BCB008  436            ST      @FAC+1,RAM(@Q2+2)                      1
64FD 4B
64FE 9108    437            DINC    @Q2+2                                  1
6500 A30400  438            DADD    8,@Q1+2                                1
6503 08
6504 910A    439            DINC    @Q3                                    1
6506 0564EC  440            $END
6509 060012  441            CALL    RPL
             442  *
650C AE0608  443  BYTBIT    DIV     8,@Q2
650F 8608    444            CLR     @Q2+2
6511 BC0906  445            ST      @Q2,@Q2+3
6514 8606    446            CLR     @Q2
6516 A10408  447            DADD    @Q2+2,@Q1+2
6519 00      448            RTN
             449  *
651A 663065  450  RUN       DATA    #SAVE,#RUN01
651D 22
651E 035255  451            DATA    3,:RUN:
6521 4E
6522 D7703F  452  RUN01     $IF     @PTOP .DEQ. >3FFF GOTO RUN02
6525 FF658A
6528 BD0270  453            DST     @PTOP,@DEST
652B 9102    454            DINC    @DEST
652D BD0002  455            DST     @DEST,@SRC
6530 BF083F  456            DST     >3FFF,@BLEN
6533 FF
6534 A50870  457            DSUB    @PTOP,@BLEN
6537 BF703F  458            DST     >3FFF,@PTOP
653A FF
653B A50008  459            DSUB    @BLEN,@SRC
653E C90030  460            $WHILE  @SRC .DHE. @STLN                       1
6541 4552
6543 3408B0  461            MOVE    @BLEN FROM RAM(@SRC) TO RAM(@DEST)     1
6546 02B000
```

```
6549 BD0200  462          DST    @SRC,@DEST                                  1
654C A50008  463          DSUB   @BLEN,@SRC                                  1
654F 05653E  464          $END
6552 BD0A02  465          DST    @DEST,@03
6555 A50A30  466          DSUB   @STLN,@03
6558 BD0230  467          DST    @STLN,@DEST
655B A10208  468          DADD   @BLEN,@DEST
655E 340AB0  469          MOVE   @03 FROM RAM(@STLN) TO RAM(@DEST)
6561 02B030
6564 A13208  470          DADD   @BLEN,@ENLN
6567 A13008  471          DADD   @BLEN,@STLN
656A BD0A30  472          DST    @STLN,@03
656D 950A    473          DINCT  @03              POINTER TO LINE ADDR
656F D10A32  474          $WHILE @03 .DLT. @ENLN                             1
6572 658A
6574 350002  475          MOVE   2 FROM RAM(@03) TO @0                       1
6577 00B00A
657A A10008  476          DADD   @BLEN,@0                                    1
657D 350002  477          MOVE   2 FROM @0 TO RAM(@03)                       1
6580 B00A00
6583 A30A00  478          DADD   4,@03                                       1
6586 04
6587 05656F  479          $END
658A BD3432  480  RUN02   DST    @ENLN,@DATA       SET-UP FOR FIRST LINE
658D A73400  481          DSUB   >03,@DATA
6590 03
6591 87A3EC  482          DCLR   RAM(SAVEPC)       DISALLOW CONTINUE
6594 BF6E05  483          DST    VRAMVS,@VSPTR     SET STACK EMPTY
6597 F8
6598 BD246E  484          DST    @VSPTR,@STVSPT
659B BE4401  485          ST     1,@BUFFY
659E 052840  486          B      PRGLST
             487  * CIF  -CONVERT INTEGER TO FLOATING    -
             488  *      -INTEGER IN @>00               -
             489  *      -FLOATING IN FAC,FAC+1,....FAC+7-
     65A1    490  CIF      EQU    $
65A1 874A    491           DCLR   @FAC
65A3 BD4C00  492           DST    @0,@FAC+2
65A6 874E    493           DCLR   @FAC+4
65A8 8750    494           DCLR   @FAC+6
65AA 8F4C    495           DCZ    @FAC+2
65AC 65CE    496           BS     CIFEND
65AE AF4A27  497           DDIV   10000,@FAC
65B1 10
65B2 AE4C64  498           DIV    100,@FAC+2
65B5 BE4A42  499           ST     >42,@FAC
65B8 8E4B    500           CZ     @FAC+1
65BA 45CE    501           BR     CIFEND
65BC 350003  502           MOVE   3 FROM @FAC+2 TO @FAC+1
65BF 4B4C
65C1 A64A01  503           SUB    1,@FAC
65C4 8E4B    504           CZ     @FAC+1
65C6 45CE    505           BR     CIFEND
65C8 BD4B4C  506           DST    @FAC+2,@FAC+1
65CB A64A01  507           SUB    1,@FAC
```

```
     65CE    508  CIFEND   EQU    $
65CE 00      509           RTN
             510  *
             511  ***  CONVERT FLOATING TO INTEGER
             512  ***  PLACE INTEGER IN FLOATING(LOW WORD)
             513  ***       MARK BITS 32(INTEGER PRESENT)
             514  ***                 40(FLOATING PRESENT)
             515  *
     65CF    516  CVFLIN   EQU    $
65CF DA4E80  517           CLOG   >80,@FAC+4        IS INTEGER PRESENT?
65D2 4609    518           BR     CVEND             YES, SHORT-CUT
65D4 874E    519           DCLR   @FAC+4            NO,  CALCULATE INTEG
65D6 8750    520           DCLR   @FAC+6
65D8 CE4A42  521           $IF    @FAC .GT. >42 GOTO CVEND
65DB 6609
65DD D24A40  522           $IF    @FAC .LT. >40 GOTO CVEND
65E0 4609
65E2 BC514B  523           ST     @FAC+1,@FAC+7
65E5 D64A40  524           $IF    @FAC .EQ. >40 GOTO CVEND
65E8 6609
65EA BC4F4C  525           ST     @FAC+2,@FAC+5
65ED AB5000  526           DMUL   100,@FAC+6
65F0 64
65F1 BD5052  527           DST    @FAC+8,@FAC+6
65F4 A1504E  528           DADD   @FAC+4,@FAC+6
65F7 064A41  529           $IF    @FAC .EQ. >41 GOTO CVEND
65FA 6609
65FC BC4F4D  530           ST     @FAC+3,@FAC+5
65FF A85000  531           DMUL   100,@FAC+6
6602 64
6603 BD5052  532           DST    @FAC+8@FAC+6
6606 A1504E  533           DADD   @FAC+4,@FAC+6
6609 BD4A50  534  CVEND    DST    @FAC+6,@FAC            MOVE INTEGER TO FAC
660C 00      535           RTN
     660D    536  ERR7     EQU    $
660D 06001C  537           CALL   ERR$$
6610 6612    538           DATA   #SYNERR
             539           BASE   0,0,>300,>300,0,0,>60
6612 128A8A  540  SYNERR   DATA   18,:** SYNTAX ERROR **:
6615 80B3B9
6618 AEB4A1
661B B880A5
661E B2B2AF
6621 B2808A
6624 8A
             541           BASE   0,0,>300,>300,0,0,
6625 8E8089  542  PGMCTR   CZ     @>89
6628 462D    543           BR     PGMCT1
662A BD2C56  544           DST    @>56,@>2C
662D 0F1B    545  PGMCT1   XML    PGMCHR
662F 00      546           RTN
6630 665E66  547  SAVE     DATA   #LOAD,#SAVE01
6633 3A
6634 054053  548           DATA   5,:@SAVE:
6637 415645
```

```
                 ***************************************************
                 **** CALL @SAVE(LEN,ADDR,NAME-LEN,NAME-ADDR) *****
                 ***************************************************
663A 066125  SAVE01   CALL   P
663D BEA3C0          ST     6,RAM(R0L0UT)          CODE FOR 'SAVE'
6640 06
6641 BDA3C6  SAVE02   DST    @01,RAM(R0L0UT+6)      LENGTH
6644 02
6645 BDA3C2          DST    @02,RAM(R0L0UT+2)      ADDRESS
6648 06
6649 BF5603          DST    R0L0UT+9,@PABL0C       DSR PTR
664C C9
664D 340AA3          MOVE   @03 FROM RAM(@04) TO RAM(R0L0UT+10)     FILE NA
6650 CAB00E
                 *  03 MUST BE LESS THAN 18
6653 BCA3C9          ST     @03+1,RAM(R0L0UT+9)    NAME LENGTH
6656 0B
6657 060010          CALL   CALDSR                 CALL DSR
665A 08              DATA   8
665B 060012          CALL   RPL                    RETURN TO BASIC
665E 667266  LOAD    DATA   #EXIT,#LOAD01
6661 68
6662 05404C          DATA   5,:@LOAD:
6665 4F4144
6668 066125  LOAD01  CALL   P
666B BEA3C0          ST     5,RAM(R0L0UT)          CODE FOR 'LOAD'
666E 05
666F 056641          B      SAVE02
6672 66D466  EXIT    DATA   #SASUB,#EXIT01
6675 7C
6676 054045          DATA   5,:@EXIT:
6679 584954
667C 0B      EXIT01  EXIT
667D 6698    PROG1   DATA   #PROG2            PTR TO NEXT
667F 6696            DATA   #PROG11           ENTRY
6681 144350          DATA   20,:CPU PROG AT C(>6000):
6684 552050
6687 524F47
668A 204154
668D 204328
6690 3E3630
6693 303029
6696 0F70    PROG11  XML    >70               CPU >6010
6698 66BB    PROG2   DATA   #PROG3            PTR TO NEXT
669A 66B1            DATA   #PROG21           ENTRY
669C 144350          DATA   20,:CPU PROG AT G(>7000):
669F 552050
66A2 524F47
66A5 204154
66A8 204728
66AB 3E3730
66AE 303029
66B1 311000  PROG21  MOVE   >2000 FROM ROM(#>7000) TO @>9D00      [2000-830
66B4 8F9D00
66B7 7000
```

```
66B9 0F20    579            XML     >20                CPU >2000
             580   *
66BB 0000    581   PROG3    DATA    #0
66BD 7000    582            DATA    #>7000
66BF 144750  583            DATA    20,:GPL PROG AT G(>7000):
66C2 4C2050
66C5 524F47
66C8 204154
66CB 204728
6CE  3E3730
6601 303029
             584   *        END
```

```
                 *        Use of CPU ram locations
                 *
0000             PTFBSL   EQU    >00        Pointer To First Byte Speak List
0002             PTLBSL   EQU    >02        Pointer To Last Byte Speak List
0004             PTEBSL   EQU    >04        Pointer To End Byte Speak List
0000             PHLEN    EQU    >00        PHrom Data Length
0001             PHRADD   EQU    >01        PHRom ADDress

0006             PTFCIS   EQU    >06        Pointer To First Char In String
0008             PTCCIS   EQU    >08        Pointer To Current Char In String
000A             PTLCIS   EQU    >0A        Pointer To Last Char In String

000C             PTFCIP   EQU    >0C        Pointer To First Char In Phrase
000E             PTCCIP   EQU    >0E        Pointer To Current Char In Phrase
0010             PTLCIP   EQU    >10        Pointer To Last Char In Phrase

0012             PTFBPH   EQU    >12        Pointer To First Byte In PHrom
0014             PTCCPH   EQU    >14        Pointer To Current Char In PHrom
0016             PTLCPH   EQU    >16        Pointer To Last Char In PHrom

004A             CCHAR    EQU    >4A        Current CHAR
004B             TOTTIM   EQU    >4B        TOTal wait TIMe
004C             SPELFLG  EQU    >4C        SPelL out phrase FLaG
004D             DATAAD   EQU    >4D        speech DATA ADdr
004F             PTLCIL   EQU    >4F        Pointer To Last Char In List
0051             TIMLEN   EQU    >51        TIMe LENgth of timing char
0052             PHADDR   EQU    >52        PHrom ADDRess

0054             TEMP1    EQU    >54        TEMPorary cpu location 1
0056             TEMP2    EQU    >56        TEMPorary cpu location 2

0058             READ     EQU    >58        Contains addr of speech peripheral
                 *                               READ byte interface
005A             WRITE    EQU    >5A        Contains addr of speech peripheral
                 *                               WRITE byte interface
005C             COND     EQU    >5C        CONDition flag used in testing
005D             PHDATA   EQU    >5D        PHrom DATA
005E             PTCBED   EQU    >5E        Pointer To Current Byte Ext Data
0060             PTLBED   EQU    >60        Pointer To Last Byte Ext Data
0062             PTEBED   EQU    >62        Pointer To End Byte Ext Data
0064             STRLEN   EQU    >64        STRing LENgth
0066             BYTE1    EQU    >66        BYTE 1
0067             BYTE2    EQU    >67        BYTE 2
0068             BYTE3    EQU    >68        BYTE 3
0069             SPKSTS   EQU    >69        SPeaK StaTuS
```

```
                 632  ****************************************************************
                 633  * ASCII symbol definitions
                 634
0041             635  ASCIIA    EQU     :A:
0029             636  SPACE     EQU     : :
0023             637  NUMBER    EQU     :#:
003B             638  SEMICO    EQU     :;:
003A             639  COLON     EQU     ::::
002E             640  PERIOD    EQU     :.:
002B             641  PLUS      EQU     :+:
002C             642  COMMA     EQU     :,:
002D             643  HYPEN     EQU     :-:
0030             644  ZERO      EQU     :0:
0039             645  NINE      EQU     :9:
                 646  ****************************************************************
                 647  * External BASIC subroutine and other standard definitions
0017             648  VPUSH     EQU     23          Push value onto the stack
0018             649  VPOP      EQU     24          Pop value off the stack
0010             650  CPL       EQU     >10         Call link
0038             651  GETSTR    EQU     >38         Get string
0079             652  TIMER     EQU     >79         Location of interval timer
0089             653  GROMFL    EQU     >89         Grom flag
002C             654  CPU2C     EQU     >2C
001C             655  SREF      EQU     >1C         String reference
                 656  ****************************************************************
```

```
66D4 66DC6A  658  SASUB     DATA    #SPSUB,#SAY,3,:SAY:
66D7 280353
66DA 4159
66DC 68D26A  659  SPSUB     DATA    #ENDLNK,#SPGET,5,:SPGET:
66DF 9F0553
66E2 504745
66E5 54
             660  MAKESL
66E6 066980  661            CALL    SETRW
66E9 BD0806  662            DST     @PTFCIS,@PTCCIS
66EC 0667BD  663            CALL    GETTIM
66EF 864B    664            CLR     @TOTTIM
66F1 0667AD  665            CALL    TIMING
66F4 C5080A  666            $WHILE  @PTCCIS .NOT. .DH. @PTLCIS                    1
66F7 6748
66F9 066752  667            CALL    PHRASE                                        1
66FC 8E4C47  668            $IF     @SPLFLG .EQ. 0 THEN                           2
66FF 11
6700 066859  669            CALL    LOOKUP                                        2
6703 8F4D47  670            $IF     @DATAAD .DEQ. 0 THEN                          3
6706 0C
6707 BE4C01  671            ST      >1,@SPLFLG                                    3
670A 4711    672            $SELSE                                                3
670C 066968  673            CALL    STDATA                                        3
670F 864B    674            CLR     @TOTTIM                                       3
             675            $END
             676            $END
6711 D64C01  677            $IF     @SPLFLG .EQ. 1 THEN                           2
6714 4743
6716 BD4F10  678            DST     @PTLCIP,@PTLCIL                               2
6719 BE4B3C  679            ST      >3C,@TOTTIM                                   2
             680            $REPEAT
671C D6B00C  681            $WHILE  RAM(@PTFCIP) .EQ. SPACE                       4
671F 204729
6722 BE4B3C  682            ST      >3C,@TOTTIM                                   4
6725 910C    683            DINC    @PTFCIP                                       4
6727 471C    684            $SEND                                                 3
6729 BD100C  685            DST     @PTFCIP,@PTLCIP                               3
672C 066859  686            CALL    LOOKUP                                        3
672F 8F4D47  687            $IF     @DATAAD .DEQ. 0 THEN                          4
6732 37
6733 BF4D71  688            DST     >71F4,@DATAAD   Put addr of 'UHOH' in DATAAD  4
6736 F4
             689            $END
6737 066968  690            CALL    STDATA                                        3
673A 864B    691            CLR     @TOTTIM                                       3
673C 910C    692            DINC    @PTFCIP                                       3
673E C50C4F  693            $UNTIL  @PTFCIP .DH. @PTLCIL                          2
6741 471C
             694            $END
6743 0667AD  695            CALL    TIMING                                        1
6746 46F4    696            $SEND
6748 8E4B67  697            $IF     @TOTTIM .NE. 0 THEN                           1
674B 51
674C 874D    698            DCLR    @DATAAD                                       1
```

```
674E 066968  699          CALL    STDATA                                    1
             700          $END
6751 00      701          RTN
```

```
                703  PHRASE
6752 D64A23     704          $IF     @CCHAR .EQ. NUMBER THEN   Do we have a phrase? 1
6755 478E
6757 9108       705          DINC    @PTCCIS                   You bet we do!       1
6759 D6B008     706          $WHILE  RAM(@PTCCIS) .EQ. SPACE                        2
675C 304764
675F 9108       707          DINC    @PTCCIS                                        2
6761 056759     708          $END                                                   1
6764 D6B008     709          $IF     RAM(@PTCCIS) .EQ. NUMBER THEN  Null phrase     2
6767 2347 6D                                                        (all spaces)
676A 9108       710          DINC    @PTCCIS     Yes, Go on to next char            2
676C 00         711          RTN                 Ignore this phrase, go to next one 2
                712          $END
676D BD0C08     713          DST     @PTCCIS,@PTFCIP                                1
                714          $REPEAT
6770 9108       715          DINC    @PTCCIS                                        2
6772 BC4AB0     716          ST      RAM(@PTCCIS),@CCHAR                            2
6775 08
6776 C5080A     717          $IF     @PTCCIS .DH. @PTLCIS GOTO FNDNUM               2
6779 6780
677B D64A23     718          $UNTIL  @CCHAR .EQ. NUMBER                             1
677E 4770
                719  FNDNUM
6780 BD1008     720          DST     @PTCCIS,@PTLCIP                                1
6783 9310       721          DDEC    @PTLCIP                                        1
6785 9108       722          DINC    @PTCCIS     Point to char after #              1
6787 0667BD     723          CALL    GETTIM                                         1
678A 864C       724          CLR     @SPLFLG     Clear spell-flag (don't spell)     1
678C 47AC       725          $SELSE                                                 1
678E BD0C08     726          DST     @PTCCIS,@PTFCIP                                1
6791 CA4A41     727          $IF     @CCHAR .HE. ASCIIA THEN   1st char alphabetic? 2
6794 479A
6796 864C       728          CLR     @SPLFLG    Yes, clear spell-flag (don't spell) 2
6798 479D       729          $SELSE                                                 2
679A BE4C01     730          ST      >1,@SPLFLG  No, set spell-flag (do spell)      2
                731          $END
                732          $REPEAT
679D 9108       733          DINC    @PTCCIS                                        2
679F 0667BD     734          CALL    GETTIM                                         2
67A2 D651FF     735          $UNTIL  @TIMLEN .NE. >FF  Ptcc will point to following 1
67A5 679D
67A7 BD1008     736          DST     @PTCCIS,@PTLCIP                                1
67AA 9310       737          DDEC    @PTLCIP                                        1
                738          $END
67AC 00         739          RTN
```

```
                741  TIMING
67AD CA51FE     742          $WHILE  @TIMLEN .NOT. .HE. >FE                        1
67B0 67BC
67B2 A14B51     743          DADD    @TIMLEN,@TOTTIM                               1
67B5 9108       744          DINC    @PTCCIS                                       1
67B7 0667BD     745          CALL    GETTIM                                        1
67BA 47AD       746          $SEND
67BC 00         747          RTN
                748
                749  GETTIM
67BD BC4AB0     750          ST      RAM(@PTCCIS),@CCHAR
67C0 08
67C1 B05408     751          DST     @PTCCIS,@TEMP1
67C4 C5540A     752          $IF     @TEMP1 .DH. @PTLCIS THEN                      1
67C7 47CD
67C9 BE51FE     753          ST      >FE,@TIMLEN                                   1
67CC 00         754          RTN                                                   1
                755          $END
67CD C64A3B     756          $IF     @CCHAR .NOT. .H. SEMICO THEN                  1
67D0 682F
67D2 D64A20     757          $IF      @CCHAR .EQ. SPACE THEN                       2
67D5 47DE
67D7 BE5106     758          ST      >06,@TIMLEN                                   2
67DA 06684D     759          CALL    SKPSPC            Skip over all extra spaces  2
67DD 00         760          RTN                                                   2
                761          $END
67DE D64A2B     762          $IF     @CCHAR .EQ. PLUS THEN                         2
67E1 47ED
67E3 9154       763          DINC    @TEMP1            Need to test the next char  2
67E5 066833     764          CALL    NUMERC            Is it numeric?              2
67E8 682F       765          BS      NOTIME  Was numeric, therefore not timing char2
67EA 8651       766          CLR     @TIMLEN           No, set as notiming         2
67EC 00         767          RTN                                                   2
                768          $END
67ED D64A2C     769          $IF     @CCHAR .EQ. COMMA THEN                        2
67F0 47F6
67F2 BE5112     770          ST      >12,@TIMLEN                                   2
67F5 00         771          RTN                                                   2
                772          $END
67F6 D64A2E     773          $IF     @CCHAR .EQ. PERIOD THEN                       2
67F9 480D
67FB 9354       774          DDEC    @TEMP1            Go back to preceding char   2
67FD 066833     775          CALL    NUMERC            Is it numeric?              2
6800 4809       776          BR      PTIME                                         2
6802 9554       777          DINCT   @TEMP1            Yes, on to following char   2
6804 066833     778          CALL    NUMERC            Is it numeric too?          2
6807 682F       779          BS      NOTIME  Both were numeric, therefore not a tim2
6809 BE513C     780  PTIME   ST      >3C,@TIMLEN  Both were not numeric therefore a2
680C 00         781          RTN                                                   2
                782          $END
680D D64A2D     783          $IF     @CCHAR .EQ. HYPEN THEN                        2
6810 481D
6812 9154       784          DINC    @TEMP1            Check next char             2
6814 066833     785          CALL    NUMERC            Is it numeric?              2
6817 682F       786          BS      NOTIME  Was numeric, therefore not a timing ch2
```

```
6819 BE510C  787          ST     >0C,@TIMLEN   Was not numeric, therefore set a2
681C 00      788          RTN                                                  2
             789          $END
681D D64A3A  790          $IF    @CCHAR .EQ. COLON THEN                        2
6820 4826
6822 BE5130  791          ST     >30,@TIMLEN                                   2
6825 00      792          RTN                                                  2
             793          $END
6826 D64A3B  794          $IF    @CCHAR .EQ. SEMICO THEN                       2
6829 482F
682B BE511E  795          ST     >1E,@TIMLEN                                   2
682E 00      796          RTN
             797          $END
             798          $END
             799  NOTIME
682F BE51FF  800          ST     >FF,@TIMLEN    Set as no timing char present
6832 00      801          RTN
```

```
                 803  *  This routine tests the char pointed to by PTCCIS and detects
                 804  *       1- it is within the current string boundaries
                 805  *       2- it is numeric (is between '0' and '9')
                 806  *  If both of the above conditions are true, COND is set upo
                 807  *  Otherwise, COND will be reset
                 808  NUMERC
6833 C5540A      809            $IF     @TEMP1 .NOT. .DH. @PTLCIS THEN                1
6836 684C
6838 C50654      810            $IF     @PTFCIS .NOT. .DH. @TEMP1 THEN                2
683B 684C
683D CAB054      811            $IF     RAM(@TEMP1) .HE. @TEMP1 THEN                  3
6840 30484C
6843 C6B054      812            $IF     RAM(@TEMP1) .NOT. .H. NINE THEN               4
6846 39684C
6849 D40101      813            CEQ     @01,@01           Set condition bit to on     4
                 814            $END
                 815            $END
                 816            $END
                 817            $END
684C 01          818            RTNC
                 819
                 820  SKPSPC
684D D6E001      821            $WHILE  RAM(1(PTCCIP)) .EQ. SPACE                     1
6850 0E2048
6853 58
6854 910E        822            DINC    @PTCCIP                                       1
6856 484D        823            $SEND
6858 00          824            RTN
                 825
                 826  LOOKUP
6859 8766        827            DCLR    @BYTE1
                 828  TRYAGN
685B BD5266      829            DST     @BYTE1,@PHADDR   Check for >AA bd Phrom heading
685E 06691E      830            CALL    LOADAD
6861 BEC000      831            ST      >10,@0(WRITE)
6864 5A10
6866 D6C000      832            $IF     @0(READ) .EQ. >AA THEN  Is there a Phrom there1
6869 58AA48
686C 79
686D BD1266      833            DST     @BYTE1,@PTFBPH    Yes, prepare to search it fro
6870 9112        834            DINC    @PTFBPH           Just one beyond the header  1
6872 066884      835            CALL    SEARCH                                        1
6875 8F4D48      836            $IF     @DATAAD .DNE. 0 GOTO FOUND                    1
6878 83
                 837            $END
6879 A36680      838            DADD    >8000,@BYTE1
687C 00
687D 8F6648      839            $IF     @BYTE1 .DNE. 0 GOTO TRYAGN
6880 5B
6881 874D        840            DCLR    @DATAAD
                 841  FOUND
6883 00          842            RTN
```

```
                844  SEARCH
6884 BD5212     845          DST    @PTFBPH,@PHADDR  LOADAD must hav addr in PHADDR
6887 06691E     846          CALL   LOADAD      Set up the Phrom to scan a word
688A BEC000     847          ST     >10,@0(WRITE)
688D 5A10
688F 8616       848          CLR    @PTLCPH     Save the length of the word in 2 by
6891 BC17C0     849          ST     @0(READ),@PTLCPH+1    at PTLCPH
6894 0058
6896 A11612     850          DADD   @PTFBPH,@PTLCPH   Then add front pointer giving
6899 BD1412     851          DST    @PTFBPH,@PTCCPH   Set up current byte pointer i
689C 9114       852          DINC   @PTCCPH     1 byte beyond the length byte is fi
689E BD0E0C     853          DST    @PTFCIP,@PTCCPH   Reset current byte pointer in
                854  *   Try comparing chars
                855  NEXT
68A1 BEC000     856          ST     >10,@0(WRITE)     Read a byte from Phrom
68A4 5A10
68A6 BC5DC0     857          ST     @0(READ),@PHDATA   Get data in from the Phrom
68A9 0058
68AB D45DB0     858          $IF    @PHDATA .EQ. RAM(@PTCCIP) THEN   Compare Phrom
68AE 0E48FD
68B1 9114       859          DINC   @PTCCPH           Equal, advance both pointers1
68B3 910E       860          DINC   @PTCCIP                                        1
68B5 D6B00E     861          $IF    RAM(@PTCCIP) .EQ. SPACE THEN   Need to skip ext2
68B8 2048C7
68BB 06684D     862          CALL   SKPSPC                                         2
68BE D6E001     863          $IF    RAM(1(PTCCIP)) .EQ. NUMBER THEN                3
68C1 0E2348
68C4 C7
68C5 910E       864          DINC   @PTCCIP                                        3
                865          $END
                866          $END
68C7 C51416     867          $IF    @PTCCPH .DH. @PTLCPH THEN  Scanned all char in2
68CA 48F0
68CC C50E10     868          $IF    @PTCCIP .DH. @PTLCIP THEN  Yes, scanned all ch3
68CF 48E8
68D1 BD5216     869          DST    @PTLCPH,@PHADDR    Yes, word found            3
68D4 A35200     870          DADD   >6,@PHADDR   Move 5 bytes down from last char 3
68D7 06
68D8 066950     871          CALL   READAD   This will set up data addr in DATADD 3
68DB BEC000     872          ST     >10,@0(WRITE)                                  3
68DE 5A10
68E0 BC64C0     873          ST     @0(READ),@STRLEN                               3
68E3 0058
68E5 00         874          RTN                                                   3
68E6 48EE       875          $SELSE  No. Phrase is longer, follow the higher link 3
68E8 BF5200     876          DST    >3,@PHADDR  Move 3 bytes beyond PTLCPH to get 3
68EB 03
68EC 490D       877          BR     NXTPHR                                         3
                878          $END
68EE 48FB       879          $SELSE                                                2
68F0 C50E10     880          $IF    @PTCCIP .NOT. .DH. @PTLCIP GOTO NEXT  Go compa2
68F3 48A1
68F5 BF5200     881          DST    >1,@PHADDR  Move 1 byte beyond PTLCPH to get  2
68F8 01
68F9 490D       882          BR     NXTPHR                                         2
```

```
                883          $END
68FB 491E       884          $ELSE                                          1
68FD C45DB0     885          $IF    @PHDATA .H. RAM(@PTCCIP) THEN  Phrom word is b2
6900 0E4909
6903 BF5200     886          DST    >1,@PHADDR   Go 1 byte beyond last char to low 2
6906 01
6907 490D       887          $ELSE    Phrase word comes before Phrom word        2
6909 BF5200     888          DST    >3,@PHADDR   Go 3 bytes beyond last char to hi2
690C 03
                889          $END
                890  NXTPHR
690D A15216     891          DADD   @PTLCPH,@PHADDR                            1
6910 066950     892          CALL   READAD       Now get the new addr           1
6913 8F4D49     893          $IF    @DATAAD .DEQ. #0 THEN  Any more leaves on this2
6916 18
6917 00         894          RTN                No, return empty handed         2
                895          $END               Yes, set up to check next word
6918 BD124D     896          DST    @DATAAD,@PTFBPH   Store new addr in PTFBPH    1
691B 056884     897          B      SEARCH            Go compare this new word!   1
                898          $END
                899  * The program cannot reach this point!!
```

```
                901  LOADAD
691E BD5452     902          DST    @PHADDR,@TEMP1    This is destructive, so copy
6921 BD5652     903          DST    @PHADDR,@TEMP2     address into temporary areas
6924 E65404     904          SRL    @TEMP1,4          Isolate the MSN of the MSB
6927 E65504     905          SRL    @TEMP1+1,4        Isolate the MSN of the LSB
692A B360F      906          DAND   >0F0F,@TEMP2   Isolate LSN of both the MSB, LSB
692D 0F
692E B75440     907          DOR    >4040,@TEMP1   Include a 4 as MSN of all four
6931 40
6932 B75640     908          DOR    >4040,@TEMP2   to indicate a Load Address Cmd
6935 40
6936 BCC000     909          ST     @TEMP2+1,@0(WRITE)   Write out LSN of the LSB
6939 5A57
693B BCC000     910          ST     @TEMP1+1,@0(WRITE)   Write out LSN of the MSB
693E 5A55
6940 BCC000     911          ST     @TEMP2,@0(WRITE)  Write out the MSN of the LSB
6943 5A56
6945 BCC000     912          ST     @TEMP1,@0(WRITE)  Write out the MSN of the MSB
6948 5A54
694A BEC000     913          ST     >40,@0(WRITE)     Write out 0 as fifth nybble
694D 5A40
694F 00         914          RTN
                915
                916  READAD
6950 06691E     917          CALL   LOADAD  PHADDR should be set up with addr in it
6953 BEC000     918          ST     >10,@0(WRITE)         Get high byte of addr
6956 5A10
6958 BC4DC0     919          ST     @0(READ),@DATAAD      Store it in DATAAD
695B 0058
695D BEC000     920          ST     >10,@0(WRITE)         Get low byte of addr
6960 5A10
6962 BC4EC0     921          ST     @0(READ),@DATAAD+1    Store it in DATAAD+1
6965 0058
6967 00         922          RTN
                923
                924  STDATA
6968 D50204     925          $IF    @PTLBSL .EQ. @PTEBSL GOTO ERR6
696B 6BCD
696D BCB002     926          ST     @TOTTIM,RAM(@PTLBSL)
6970 4B
6971 9102       927          DINC   @PTLBSL
6973 BCB002     928          ST     @DATAAD,RAM(@PTLBSL)
6976 4D
6977 9102       929          DINC   @PTLBSL
6979 BCB002     930          ST     @DATAAD+1,RAM(@PTLBSL)
697C 4E
697D 9102       931          DINC   @PTLBSL
697F 00         932          RTN
                933
                934  SETRW
6980 310004     935          MOVE   4 FROM ROM(>46) TO @READ
6983 580046
6986 00         936          RTN
```

```
                938  SPEAK
6987 066980     939          CALL    SETRW
698A C90002     940          $WHILE  @PTFBSL .NOT. .DHE. @PTLBSL                    1
698D 6A13
698F 066A1F     941          CALL    WAIT                                           1
6992 D6B000     942          $IF     RAM(@PTFBSL) .NE. >FF THEN                     2
6995 FF69BA
6998 BC79B0     943          ST      RAM(@PTFBSL),@TIMER                            2
699B 00
699C 8279       944          NEG     @TIMER                                         2
699E BD52E0     945          DST     RAM(1(@PTFBSL)),@PHADDR                        2
69A1 0100
69A3 A30000     946          DADD    3,@PTFBSL                                      2
69A6 03
69A7 D27900     947  LOOP1   $IF     @TIMER .LT. 0 GOTO LOOP1                       2
69AA 49A7
69AC 8E5269     948          $IF     @PHADDR .NE. 0 THEN                            3
69AF B8
69B0 06691E     949          CALL    LOADAD                                         3
69B3 BEC000     950          ST      >50,@0(WRITE)                                  3
69B6 5A50
                951          $END
69B8 4A10       952          $ELSE                                                  2
69BA 9100       953          DINC    @PTFBSL                 Skip >FF               2
69BC BD5EB0     954          DST     RAM(@PTFBSL),@PTCBED   Set up pointer to front2
69BF 00
69C0 D6B05E     955          $IF     RAM(@PTCBED) .NE. >60 GOTO ERR1  Invalid speec2
69C3 604BBE
69C6 8662       956          CLR     @PTEBED                                        2
69C8 BC63EF     957          ST      RAM(-1(PTCBED)),@PTEBED+1   Get total length o2
69CB FFFF5E
69CE A1625E     958          DADD    @PTCBED,@PTEBED   Length+first byte=last byte 2
69D1 9500       959          DINCT   @PTFBSL           Skip addr bytes              2
                960  DIRSPH
69D3 BEC000     961          ST      >60,@0(WRITE)    Start Speak External          2
69D6 5A60
69D8 915E       962          DINC    @PTCBED          Skip speak external command   2
69DA BD6080     963          DST     RAM(@PTCBED),@PTLBED   Set up pointer to last 2
69DD 5E
69DE 955E       964          DINCT   @PTCBED   Skip length bytes, this is first rea 2
69E0 A1605E     965          DADD    @PTCBED,@PTLBED   Add length to first byte giv 2
69E3 BE5610     966          ST      16,@TEMP2                                      2
69E6 D55E62     967  LOOPR   $IF     @PTCBED .DNE. @PTEBED GOTO CONTIN              2
69E9 6A10
69EB D55E60     968          $IF     @PTCBED .DEQ. @PTLBED THEN                     3
69EE 49F6
69F0 066A1F     969          CALL    WAIT                                           3
69F3 0569D3     970          B       DIRSPH                                         3
                971          $END
69F6 BCC000     972          ST      RAM(@PTCBED),@0(WRITE)                         2
69F9 5AB05E
69FC 915E       973          DINC    @PTCBED                                        2
69FE 9256       974          DEC     @TEMP2                                         2
6A00 49E6       975          BR      LOOPR                                          2
                976          $REPEAT
```

```
6A02 066A14  977            CALL    GETSTS                                  3
6A05 DA6940  978            $UNTIL .BIT6 @SPKSTS .EQ. 1                     2
6A08 6A02
680A BE5608  979            ST      8,@TEMP2                                2
6A0D 0569E6  980            B       LOOPR                                   2
             981            $END
             982  CONTIN
6A10 05698A  983            $END
6A13 00      984            RTN
             985
             986
             987  GETSTS
6A14 BC69C0  988            ST      @0(READ),@SPKSTS
6A17 0058
6A19 BC69C0  989            ST      @0(READ),@SPKSTS
6A1C 0058
6A1E 00      990            RTN
             991
             992  WAIT
             993            $REPEAT
6A1F 066A14  994            CALL    GETSTS                                  1
6A22 DA6980  995            $UNTIL  .BIT7 @SPKSTS .EQ. 0
6A25 4A1F
6A27 00      996            RTN
```

```
                 998  ***************************************************************
                 999  *
                1000  *      BBBB     AAA     SSSS    III     CCC
                1001  *      B   B   A   A   S         I     C
                1002  *      BBBB    AAAAA    SSS      I     C
                1003  *      B   B   A   A       S     I     C
                1004  *      BBBB    A   A   SSSS     III     CCC
                1005  *
                1006  ********* S U B R O U T I N E   D E F I N I T I O N S *********
                1007  *
                1008  *
                1009  ***************************************************************
                1010  *  CALL SAY(........)
                1011  *      Decode given parameter(s).  Store all data first, then
                1012  *      go speak it all at once.
                1013  ***************************************************************
                1014  *
                1015  SAY
6A28 066A85 1016            CALL    GETPAR      Skip SAY header and "("
6A2B BF0C00 1017            DST     255,@0C  At 3 bytes/entry 255 bytes=85 entries.
6A2E FF
6A2F 060038 1018            CALL    GETSTR   Get the string space to be used as the
6A32 BF4A00 1019            DST     SREF,@FAC   Set up a string FAC entry
6A35 1C
6A36 BE4C65 1020            ST      >65,@FAC+2
6A39 BD4E1C 1021            DST     @SREF,@FAC+4
6A3C BF5000 1022            DST     255,@FAC+6
6A3F FF
6A40 0F17   1023            XML     VPUSH            and make it semi-permanent!
6A42 BD004E 1024            DST     @FAC+4,@PTFBSL   Set up pointers into the speak
6A45 BD024E 1025            DST     @FAC+4,@PTLBSL
6A48 BD0400 1026            DST     @PTFBSL,@PTEBSL
6A4B A30400 1027            DADD    255,@PTEBSL
6A4E FF
                1028  DIRSPK
6A4F 066A95 1029            CALL    GETPRM           Get parameter
6A52 8F506A 1030            $IF     @FAC+6 .DNE. #0 THEN  Found non-null ASCII st 1
6A55 64
6A56 BD064E 1031            DST     @FAC+4,@PTFCIS   Set up pointer to first char 1
6A59 BD0A50 1032            DST     @FAC+6,@PTLCIS   Set up pointer to last char i1
6A5C A10A06 1033            DADD    @PTFCIS,@PTLCIS   by adding length of string t1
6A5F 930A   1034            DDEC    @PTLCIS     first char in string and subtract 1
6A61 0666E6 1035            CALL    MAKESL      Go make the speak list            1
                1036            $END
                1037  *      Next item could be direct string.
6A64 0642B3 1038            $IF     @CHAT .EQ. COMMA$ THEN   Direct string present 1
6A67 4A7E
6A69 066A95 1039            CALL    GETPRM      Get next parameter                1
6A6C 8F506A 1040            $IF     @FAC+6 .DNE. #0 THEN   Found non-null direct s2
6A6F 79
6A70 BE4BFF 1041            ST      >FF,@TOTTIM   Mark TOTTIM as direct string add2
6A73 BD4D4E 1042            DST     @FAC+4,@DATAAD  Put addr of string in DATAAD  2
6A76 066968 1043            CALL    STDATA      And put this on the speak list    2
                1044            $END
6A79 D642B3 1045            $IF     @CHAT .EQ. COMMA$ GOTO DIRSPK   Now another wo1
```

```
6A7C 6A4F
             1046          $END
6A7E 066987  1047          CALL    SPEAK   Go say all that is on the speak list
6A81 0F18    1048          XML     VPOP    Get rid of the string space
6A83 4B5C    1049          BR      RETURN  Return to main program
```

```
                  ***************************************************************
                  *
                  *      Get parameter and check for string arguments
                  *
                  ***************************************************************
                  GETPAR
6A85 8E8089                 $IF     @GROMFL .EQ. 0 THEN   Working on VDP program
6A88 4A8D
6A8A BD2C56                 DST     @FAC+12,@CPU2C   and update the program point
                            $END    IF
6A8D 0F1B                   XML     PGMCHR           Get next token
6A8F D642B7                 $IF     @CHAT .NE. LPAR$ GOTO ERR2  Quit if no left par
6A92 4BC3
6A94 00                     RTN
                  GETPRM
6A95 0F1B                   XML     PGMCHR      Get next token
6A97 0EB6                   PARSE   RPAR$
6A99 D64C65                 $IF     @FAC+2 .NE. >65 GOTO ERR3  Not a string, someth
6A9C 4BC8
6A9E 00                     RTN                 Return normal
```

```
                 1070  ***************************************************************
                 1071  *
                 1072  *      SPGET subprogram.  Load speech data from external
                 1073  *      device.  Use standard file I/O.
                 1074  *
                 1075  ***************************************************************
                 1076  *
                 1077  SPGET
6A9F 066A85      1078            CALL    GETPAR      Skip header and "("
                 1079  NXTPAR
6AA2 066A95      1080            CALL    GETPRM      Get parameter
6AA5 8F506B      1081            $IF     @FAC+6 .DNE. #0 THEN  Found non-null ASCII str1
6AA8 57
6AA9 BD064E      1082            DST     @FAC+4,@PTFCIS  Set up pointer to first char i1
6AAC BD0A50      1083            DST     @FAC+6,@PTLCIS  Set up pointer to last char in1
6AAF A10A06      1084            DADD    @PTFCIS,@PTLCIS   by adding length of string t1
6AB2 930A        1085            DDEC    @PTLCIS     first char in string and subtracti1
6AB4 066980      1086            CALL    SETRW                                          1
6AB7 BD0806      1087            DST     @PTFCIS,@PTCCIS                                1
6ABA 0667BD      1088            CALL    GETTIM                                         1
6ABD 864B        1089            CLR     @TOTTIM                                        1
6ABF 0667AD      1090            CALL    TIMING                                         1
6AC2 C5080A      1091            $IF     @PTCCIS .NOT. .DH. @PTLCIS THEN                2
6AC5 6B57
6AC7 066752      1092            CALL    PHRASE                                         2
6ACA D64C01      1093            $IF     @SPLFLG .EQ. 1 THEN                            3
6ACD 4AD2
6ACF BD100C      1094            DST     @PTFCIP,@PTLCIP                                3
                 1095            $END
6AD2 066859      1096            CALL    LOOKUP                                         2
6AD5 8F4D4A      1097            $IF     @DATAAD .DEQ. 0 THEN                           3
6AD8 E0
6AD9 BF4D71      1098            DST     >71F4,@DATAAD                                  3
6ADC F4
6ADD BE6451      1099            ST      >51,@STRLEN                                    3
                 1100            $END
6AE0 BD014D      1101            DST     @DATAAD,@PHRADD                                2
6AE3 BC0064      1102            ST      @STRLEN,@PHLEN                                 2
6AE6 A20003      1103            ADD     >3,@PHLEN                                      2
6AE9 D642B3      1104            $IF     @CHAT .NE. COMMA$ GOTO ERR2                    2
6AEC 4BC3
6AEE 0F1B        1105            XML     PGMCHR                                         2
6AF0 0F13        1106            XML     SYM                                            2
6AF2 0F14        1107            XML     SMB                                            2
6AF4 0F17        1108            XML     VPUSH                                          2
6AF6 860C        1109            CLR     @0C                                            2
6AF8 BC0D00      1110            ST      @PHLEN,@0D                                     2
6AFB 060038      1111            CALL    GETSTR                                         2
6AFE 066980      1112            CALL    SETRW                                          2
6B01 BF4A00      1113            DST     SREF,@FAC                                      2
6B04 1C
6B05 BE4C65      1114            ST      >65,@FAC+2                                     2
6B08 BD4E1C      1115            DST     @SREF,@FAC+4                                   2
6B0B B650        1116            CLR     @FAC+6                                         2
6B0D BC5100      1117            ST      @PHLEN,@FAC+7                                  2
```

```
6B10 BEE000 1118            ST      >60,RAM(0(SREF))                          2
6B13 1C60
6B15 86E001 1119            CLR     RAM(1(SREF))                              2
6B18 1C
6B19 BCE002 1120            ST      @PHLEN,RAM(2(SREF))                       2
6B1C 1C00
6B1E A3E001 1121            DADD    -3,RAM(1(SREF))                           2
6B21 1CFFFD
6B24 BD5201 1122            DST     @PHRADD,@PHADDR                           2
6B27 06691E 1123            CALL    LOADAD                                    2
6B2A C60003 1124            $WHILE  @PHLEN .H. 3                              3
6B2D 4B55
6B2F BEC000 1125            ST      >10,@0(WRITE)                             3
6B32 5A10
6B34 BC68C0 1126            ST      @0(READ),@BYTE3                           3
6B37 0058
6B39 8667   1127            CLR     @BYTE2                                    3
6B3B BE5408 1128            ST      >8,@TEMP1                                 3
6B3E EB6700 1129  RNDAG     DSRC    @BYTE2,#1                                 3
6B41 01
6B42 E36600 1130            DSLL    @BYTE1,#1                                 3
6B45 01
6B46 9254   1131            DEC     @TEMP1                                    3
6B48 4B3E   1132            BR      RNDAG                                     3
6B4A BCE003 1133            ST      @BYTE1,RAM(3(SREF))                       3
6B4D 1C66
6B4F 911C   1134            DINC    @SREF                                     3
6B51 9200   1135            DEC     @PHLEN                                    3
6B53 4B2A   1136            $SEND                                             2
6B55 0F15   1137            XML     ASSGNV                                    2
            1138            $END
            1139            $END
6B57 D642B3 1140            $IF     @CHAT .EQ. COMMA$ GOTO NXTPAR
6B5A 6AA2
            1141  RETURN
6B5C D642B6 1142            $IF     @CHAT .NE. RPAR$ GOTO ERR2
6B5F 4BC3
6B61 0F1B   1143            XML     PGMCHR
6B63 060012 1144            CALL    RPL
```

```
            1146            BASE    0,0,>300,>300,0,0,>60
6B66 15A9AC 1147  ERR$1     DATA    21,:ILLEGAL SPEECH FORMAT:
6B69 ACA5A7
6B6C A1AC80
6B6F B3B0A5
6B72 A5A3A8
6B75 80A6AF
6B78 B2ADA1
6B7B B4
6B7C 13A9AE 1148  ERR$2     DATA    19,:INCORRECT STATEMENT:
6B7F A3AFB2
6B82 B2A5A3
6B85 B480B3
6B88 B4A1B4
6B8B A5ADA5
6B8E AEB4
6B90 16B3B4 1149  ERR$3     DATA    22,:STRING/NUMBER MISMATCH:
6B93 B2A9AE
6B96 A78FAE
6B99 B5ADA2
6B9C A5B280
6B9F ADA9B3
6BA2 ADA1B4
6BA5 A3A8
6BA7 16B3B0 1150  ERR$6     DATA    22,:SPEECH STRING TOO LONG:
6BAA A5A5A3
6BAD A880B3
6BB0 B4B2A9
6BB3 B4B2A9
6BB3 AEA780
6BB6 B4AFAF
6BB9 80ACAF
6BBC AEA7
            1151            BASE    0,0,>300,>300,0,0,0
            1152  ERR1
6BBE 06001C 1153            CALL    ERR$$
6BC1 6B66   1154            DATA    #ERR$1
            1155
            1156  ERR2
6BC3 06001C 1157            CALL    ERR$$
6BC6 6B7C   1158            DATA    #ERR$2
            1159
            1160  ERR3
6BC8 06001C 1161            CALL    ERR$$
6BCB 6B90   1162            DATA    #ERR$3
            1163
            1164  ERR6
6BCD 06001C 1165            CALL    ERR$$
6BD0 6BA7   1166            DATA    #ERR$6
6BD2 000066 1167  ENDLNK    DATA    #0,#EXIT
6BD5 72
            1168            END

ERRORS=         0
LENGTH= 3030 (>0BD6)
260 SYMBOLS USED
```

```
                 1          TITLE   GROM
                 2  ***                                                  *********
                 3  ************** DEBUGGER VERSION ******************************
                 4  ***                                                  *********
                 5  *EDTTOP   EQU     >FFD*->8300    TOP MEMORY FOR EDITOR(REAL GROM)
78F0             6  EDTTOP    EQU     >FBF)->8300    TOP MEMORY FOR EDITOR(DEBUGGER)
                 7  ****************************************************************
                 8  *   STANDALONE EDITOR & ASSEMBLER FOR 99/4                     *
                 9  ****************************************************************
                10  ****************************************************************
                11  *   USE GCA COMMAND ON THE FOLLOWING FILES                     *
                12  *   GCA FILE NAME              SD2.ASSM.SRC.GCA                *
                13  *                                                              *
                14  *   OUT PUT FILE NAME IS       SD2.ASSM.OBJ.GROM               *
                15  *   TO LIST WHOLE GPL CODE     SD2.ASSM.LST.GROM               *
                16  *                                                              *
                17  *   CONCATENATE LOADER TO THIS GPL OBJECT FILE                 *
                18  *                              SD2.ASSM.OBJ.GROM               *
                19  *                              SD2.ASSM.OBJ.ALUTIL             *
                20  *                OUTPUT FILE   SD2.ASSM.OBJ.ASSMBLR            *
                21  *                                                              *
                22  *   AUTHOR SUMIKO ENDO     Sept 1981                           *
                23  ****************************************************************
                24  ****
                25  *   EQUATES
                26  *
                27  *   XML EQUATES
                28  *
     0021       29  LOADX     EQU     >21         EXECUTE LOADER
     0022       30  SAVEX     EQU     >22         EXECUTE SAVE ROUTINE
     0023       31  EDITX     EQU     >23         EXECUTE EDITOR
                32  *
     0021       33  ASM       EQU     >21         EXECUTE ASSEMBLER
                34  *
     0021       35  ALSUP     EQU     >21         EXECUTE ASSEMBLY LANGUAGE CODE
     0022       36  TGOBLD    EQU     >22         TAGGED OBJECT LOADER.
                37  *
                38  *   INTERFACE WITH BASIC
                39  *
     0012       40  RPL       EQU     >0012       RETURN TO BASIC MAIN PROGRAM.
                41  *
                42  *   GENERAL EQUATES
                43  *
     8300       44  OFFSET    EQU     >8300       EQUATE FOR ACCESSING E-RAM
     9D00       45  RAMTOP    EQU     >2000-OFFSET  ONE LOCATION IN EXPANSION MEMORY
     9D00       46  IDCODE    EQU     >2000-OFFSET  ID CODE FOR EDITOR, ASSM, LOADER
     001F       47  CURSOR    EQU     >1F         CHARACTER NUMBER FOR CURSOR
     0018       48  CHAR2$    EQU     >18         CHARACTER TABLE ADDRESS IN MONITOR
     004A       49  CHAR3$    EQU     >4A         LOWER CASE CHARACTERS
     0020       50  SPACE     EQU     >20         SPACE CHARACTER.
     0010       51  DSRLNK    EQU     >10         LINK TO DSR ROUTINE.
                52  *
                53  *   EQUATES FOR PERIPHERAL ACCESS BLOCK
                54  *
     000A       55  PABLEN    EQU     >0A         PAB LENGTH. STANDARD
```

```
0000      56  SCREEN    EQU     >00        SCREEN OFFSET FOR PAB
003C      57  MAXLEN    EQU     60         MAXIMUM LENGTH OF FILE NAME
1000      58  PAB1      EQU     >1000      PAB ADDRESS USED FOR ALL FILE TRANSIT
1080      59  BUF1      EQU     >1080      BUFFER FOR PAB1
1100      60  PAB2      EQU     >1100      Second PAB
1180      61  BUF2      EQU     >1180      Second BUFFER
1200      62  PAB3      EQU     >1200      Third PAB
1280      63  BUF3      EQU     >1280      Third BUFFER
1300      64  PAB4      EQU     >1300      Fourth PAB for COPY statement
1380      65  BUF4      EQU     >1380      Buffer also used to load program file
          66  *
          67  *   CPU RAM EQUATES
          68  *
0000      69  TEMP      EQU     >00        Temporary storage area
0002      70  TEMP2     EQU     >02
0003      71  TEMP3     EQU     >03
0004      72  TEMP4     EQU     >04
          73  *
0010      74  DSR       EQU     >10        6 bytes temp loc for general load ro
0018      75  CHRBUF    EQU     >18        Char buffer for blinking cursor
          76  *
001C      77  SREF      EQU     >1C        Pointing to the PAB start address.
0001      78  FLG       EQU     01         Flag and status byte
0002      79  BUF       EQU     02         Buffer address
0004      80  LEN       EQU     04         Length of record
0005      81  CHRCNT    EQU     05         Character count
0006      82  RNM       EQU     06         Record number
0008      83  SCR       EQU     08         SCREEN OFFSET
0009      84  NLEN      EQU     09         Name length
          85  *
0020      86  LOC       EQU     >20        Location of a cursor
0022      87  ERRCOD    EQU     >22        Error ID code
0024      88  STLN      EQU     >24        Original cursor address
0026      89  ENDLN     EQU     >26        Address of the end of input
          90  *
0048      91  FLAG      EQU     >48        Flag bytes
0000      92  INSRT     EQU     0          Bit 0 for insert mode.
0000      93  VFLG      EQU     0          Bit 0 for peek or poke to VDP RAM
0001      94  TP        EQU     1          Bit 1 on = Thermal printer is used
0002      95  NOTHIN    EQU     2          Bit 2 ON = NO INPUT LINE IS ACCEPTED
0003      96  BSC       EQU     3          Bit 3 on = Called from BASIC program
          97  *COMP     EQU     4          Bit for compressed source file
0005      98  ASRC      EQU     5          Bit 5 on = Source file is specified
0006      99  AOBJ      EQU     6          Bit 6 on = Object file is specified
0007     100  ALST      EQU     7          Bit 7 on = List file is specified
         101  *
0049     102  FLAG2     EQU     >49
0000     103  LD        EQU     0          Bit 0 on = Loader is being executed
0001     104  VAR       EQU     1          Bit 1 on = Save with var 80 format
         105  *EDT      EQU     2          Bit 2 on = Error in editor
0003     106  OTHR      EQU     3          Bit 3 on = Flag for "other" program
         107  *ETOP     EQU     4          BIT 4 ON = EDITOR MEMORY WRITTEN OV
0005     108  GPL       EQU     5          Bit 5 on = GPL program is called fro
0006     109  LDUTIL    EQU     6          Bit 6 on = First load during LOAD &
         110  *
```

```
     004A     111  FAC       EQU     >4A
     004C     112  FAC2      EQU     FAC+2
     004E     113  FAC4      EQU     FAC+4
     0050     114  FAC6      EQU     FAC+6
     0054     115  FAC10     EQU     FAC+10
     0056     116  FAC12     EQU     FAC+12
     005C     117  ARG       EQU     >5C        Used in conversion of int to float
              118  *
              119  *         STANDARD EQUATES FOR GPL INTERPRETER
              120  *
     0070     121  MEMLEN    EQU     >70
     0073     122  SUBSTK    EQU     >73        Subroutine stack pointer.
     0074     123  KEYBRD    EQU     >74        Keyboard number to scan.
     0075     124  KEY       EQU     >75
     000D     125  ETR       EQU     >0D        Key code for enter.
     000F     126  BACK      EQU     >0F        Key code for going back.
     0076     127  JOYY      EQU     >76
     0077     128  JOYX      EQU     >77
     0078     129  RANDOM    EQU     >78
     0079     130  TIMER     EQU     >79
     007A     131  MPC       EQU     >7A
     007C     132  STATUS    EQU     >7C
              133  *
              134  ******
              135  *
              136            GROM    3
              137            ORG     0
6000 AA       138            DATA    >AA
6001 010100   139            DATA    1,1,0,#0,#PROG
6004 000060
6007 10
6008 00006A   140            DATA    #0,#LINK01,#0,#0
600B 520000
600E 0000
6010 000060   141  PROG      DATA    #0,#BEGIN
6013 25
6014 104544   142            DATA    16,:EDITOR/ASSEMBLER:
6017 49544F
601A 522F41
601D 535345
6020 4D424C
6023 4552
              143  ******
              144  ***
              145  ******
              146  ****
              147  ****************************************************************
              148  *         USER INTERFACE                                       *
              149  ****************************************************************
              150  ****************************************************************
              151  *         BEGIN - First entry point                            *
              152  ****************************************************************
              153  BEGIN
6025 878F9D   154            DCLR    @IDCODE    CLEAR ALL FILES IN MEMORY
6028 00
```

```
                 155  ****
                 156  *         Initialization section
                 157  *         Color >F5 (color table 0 through 12), blue screen
                 158  *         Define VDP registers, load character sets,
                 159  *         No motion and no sprites
                 160  ****
                 161  *
                 162  BEGIN2
6029 066525      163            CALL    COLOR       Load color table
602C BE7405      164            ST      5,@KEYBRD   Scan keyboard 5.
602F 03          165            SCAN                Enable keyboard for /4A
6030 8E7440      166            $IF     @KEYBRD .EQ. 0 THEN   MUST BE A 4A KEYBOARD   1
6033 3D
6034 BF4A0B      167            DST     >B00,@FAC                                     1
6037 00
6038 06004A      168            CALL    CHAR3$      Load lower case character         1
603B 4056        169            $ELSE                                                 1
603D BF4A0A      170            DST     @A00,@FAC                                     1
6040 00
6041 060018      171            CALL    CHAR2$      Load semi-lower character         1
6044 BF4A0B      172            DST     >B00,@FAC   First character definition        1
6047 00
6048 BEB04A      173  CHRLP     ST      >C0,RAM(@FAC)   Place two dots for lower case 1
604B C0
604C A34A00      174            DADD    8,@FAC                                        1
604F 08
6050 CB4A0C      175            $IF     @FAC .DL. >C00 GOTO CHRLP   Fix next character 1
6053 004048
                 176            $END    IF
6056 8674        177            CLR     @KEYBRD          Scan standard keyboard
                 178  *
6058 BF4A09      179            DST     >900,@FAC        Load small characters.
605B 00
605C 060018      180            CALL    CHAR2$
605F 86A800      181            CLR     RAM(>800)        Clean up TI logo stuff.
6062 3500FF      182            MOVE    >FF FROM RAM(>800) TO RAM(>801)
6065 A801AB
                 183  *
                 184  *
6069 390007      185            MOVE    7 FROM ROM(#VDPRG2) TO VDP(1)  SET UP VDP REGIS
606C 016A4B
                 186  *         CHECK IF ERAM EXISTS

***** ERROR IN IMMEDIATE FIELD  LINE=  187  COLUMN= 17

***** ERROR IN LINE BRANCH ADDRESS  LINE=  187
606F 000000      187            CALL    BUG1             TEST IF ERAM EXISTS.
                 188  *
6072 310010      189            MOVE    16 FROM ROM(#CDATA) TO CHAR(>1E)  LOAD CURSOR C
6075 A8F06A
6078 3B
                 190  *
                 191  ***************************************************************
                 192  *         START - Second entry point                          *
                 193  ***************************************************************
                 194  START
```

```
                195  ****
                196  *****
                197  *        Initial screen too select editor, assembler, loader
                198  ****
6079 BE737E     199           ST      >7E,@SUBSTK      Adjust subroutine stack pointer
607C 8722       200           DCLR    @ERRCOD          Clear error code
607E 8748       201           DCLR    @FLAG            Clear all flags
6080 0720       203           ALL     SPACE
                204           FMT     XPT=2,YPT=1,':* EDITOR/ASSEMBLER * :';
                205                   2^,11<,':PRESS:::',1^,27<;
                206                   ':1 TO EDIT:',1^,23<;
                207                   ':2    ASSEMBLE:',1^,19<;
                208                   ':3    LOAD AND RUN;',1^,15<;
                209                   ':4    RUN:',24<,1^;
                210                   ':5    RUN PROGRAM FILE:',6^,11<;
6082 08FF02     211                   '>A',':1981  TEXAS INSTRUMENTS:'
6116 03         213  SCAN10    SCAN
6117 4116       214           BR      SCAN10
6119 D6750F     215           $IF     @KEY .EQ. BACK THEN                            1
611C 411F
611E 0B         216           EXIT                                                   1
                217           $END
611F A67531     218           SUB     >31,@KEY                 Convert from ASCII
6122 CA7505     219           $IF     @KEY .HE. 5 GOTO SCAN10  Check input range
6125 6116
6127 8A75       220           CASE    @KEY
6129 4133       221           BR      EDIT                     Load a file
612B 4632       222           BR      ASSEM                    Save a file

LAST ERROR FOUND AT LINE   187

***** ERROR IN IMMEDIATE FIELD  LINE=  223  COLUMN= 17

***** ERROR IN BRANCH ADDRESS  LINE=  223
612D 0000       223           BR      LOAD                     Temp edit address

LAST ERROR FOUND AT LINE   223

***** ERROR IN IMMEDIATE FIELD  LINE=  224  COLUMN= 16

***** ERROR IN BRANCH ADDRESS  LINE=  224
612F 0000       224           BR      RUN                      Assemble a file
6131 4557       225           BR      OTHER                    Run other program
```

```
                *
                ****************************************************************
                *         SCREEN FOR EDITOR                                    *
                ****************************************************************
                ****
                EDIT
6133 066525             CALL    COLOR           Reload color table
6136 8748               DCLR    @FLAG
                        FMT     XPT=2,YPT=2,':* EDITOR *:',1^,22<;
                                ':PRESS:::',1^,28<;
                                ':1 TO LOAD:',1^,23<;
                                ':2    EDIT:',1^,23<;
                                ':3    SAVE:',1^,23<;
                                ':4    PRINT:',1^,22<;
6138 08FF02                     ':5    PURGE:',1^,22<
6192 8748               DCLR    @FLAG           Clear all flags
6194 BE737E             ST      >7E,@SUBSTK     Reset subroutine stack
6197 BF2000             DST     >C2,@LOC        Put a marker.
619A C2
                *
619B 03         SCAN70  SCAN
619C 419B               BR      SCAN70
619E D6750F             $IF     @KEY .EQ. BACK GOTO START
61A1 6079
61A3 A67531             SUB     >31,@KEY
61A6 CA7505             $IF     @KEY .HE. 5 GOTO SCAN70
61A9 619B
                *                               Put a marker in the right place
61AB BC0075             ST      @KEY,@TEMP
61AE AA0040             MUL     >40,@TEMP
61B1 A12000             DADD    @TEMP,@LOC
61B4 BEB020             ST      CURSOR,RAM(@LOC)
61B7 1F
                *
61B8 BC0475             ST      @KEY,@TEMP4     Keep the key information
                ****************************************************************
                *         CODE FOR PURGE INSTRUCTION                           *
                ****************************************************************
61BB D67504             $IF     @KEY .EQ. 4 THEN                                1
61BE 41E4
61C0 08A112             FMT     2^,':ARE YOU SURE(Y/N)? :'                      1
61C3 415245
61C6 20594F
61C9 552053
61CC 555245
61CF 28592F
61D2 4E293F
61D5 20FB
61D7 066679             CALL    YORN                                            1
61DA D67559             $IF     @KEY .EQ. :Y: THEN                              2
61DD 41E2
61DF 066549             CALL    CLRTOP          Clear top of memory for edito 2
                        $END    IF
61E2 4133               BR      EDIT                                            1
                        $END    IF
```

```
                 ***************************************************************
                 *         Take care of SAVE, PRINT, EDIT                      *
                 ***************************************************************
61E4 D67503                $IF     @KEY .EQ. 3 GOTO OPEN  IF PRINT, DO NOT LOAD ED
61E7 61FC
                 *
                 *   Load editor
                 *         Check if editor is already there(temporary)
61E9 078F9D                $IF     @IDCODE .DEQ. >55AA GOTO OPEN
61EC 0055AA
61EF 61FC
                 *         LOAD    EDITOR
61F1 0665B2                CALL    DOWNLD          CALL DOWN LOADER
61F4 6623                  DATA    #EDNAME
61F6 0665D4                CALL    DOWN10          LOAD IT DOWN.
61F9 066549                CALL    CLRTOP          CLEAR EDITOR INFORMATION
                 *
                 *
                 OPEN
61FC D60401                $IF     @TEMP4 .EQ. 1 GOTO EEDIT  GO TO EDITOR
61FF 6265
                 *         CHECK FOR VARIABLE 80 IN CASE OF SAVE
6201 D60402                $IF     @TEMP4 .EQ. 2 THEN      SAVE FILE             1
6204 4237
6206 08FF02                FMT     XPT=2,YPT=18,':VAR 80 FORMAT (Y/N)? :'        1
6209 FE1213
620C 564152
620F 203830
6212 20464F
6215 524D41
6218 542859
621B 2F4E29
621E 3F20FB
6221 06679F                CALL    YORN            GET YES OR NO ANSWER          1
6224 D67559                $IF     @KEY .EQ. :Y: THEN                            2
6227 422C
6229 B64902                SB      @FLAG2,VAR      SET VARIABLE FORMAT FLAG      2
                           $END    IF
622C D6750F                $IF     @KEY .EQ. BACK GOTO EDIT  GO BACK TO EDIT SCRE1
622F 6133
6231 BF2002                DST     >282,@LOC       ADJUST THE CURSOR             1
6234 82
6235 423B                  $ELSE                                                 1
6237 BF2002                DST     >222,@LOC       SPECIFY CURSOR LOCATION       1
623A 22
                           $END    IF
623B 31000A                MOVE    10 FROM ROM(#FLN) TO RAM(@LOC)
623E B02062
6241 5B
6242 A32000                DADD    >20,@LOC        CURSOR FOR FILE NAME.
6245 20
6246 BF1C10                DST     PAB1,@SREF
6249 00
624A 0663A0                CALL    DSR1            Get the file name and PAB
                 *         PAB is all cleared, so it is in open mode.
```

```
624D D60402  305          $IF    @TEMP4 .EQ. 2 GOTO ESAVE
6250 62B5
6252 D60403  306          $IF    @TEMP4 .EQ. 3 GOTO PRINT
6255 62D9
6257 8E0462  307          $IF    @TEMP4 .EQ. 0 GOTO ELOAD
625A 72
             308  ***
625B 46494C  310  FLN     DATA   :FILE NAME?:
             312  *****
```

```
                 ****
                 ************************************************************
                 *        EDITOR - should be changed later                  *
                 ************************************************************
                 EEDIT
6265 0F23        XML     EDITX           Execute editor
6267 67C4        BS      EMSG1           Error from editor
6269 390001      MOVE    1 FROM ROM(#ON) TO VDP(1)
626C 016271
626F 4133        BR      EDIT
6271 E0          ON      DATA    >E0
                 *
                 ************************************************************
                 *        ELOAD ---- routine to load a file                 *
                 ************************************************************
                 ELOAD
6272 06627B      CALL    FIXVAR      Try to open in fix and var format
                 *
                 ************************************************************
                 *        LOADER FOR SOURCE FILES                           *
                 ************************************************************
                 ***
                 * Note:   End of file will be checked when I/O error happens
                 *         while reading a record passed end-of-file.
                 *         See ERRIO section to see how load gets finished.
                 ****
                 READ
                 *        FIXVAR routine places 02 in PAB area for read.
                 *        ST      02,RAM(@SREF)   Ready to read a record.
6275 0F21        XML     LOADX           XML to load routine in editor
6277 67C7        BS      EMSG            Loader error
6279 4133        BR      EDIT            Finished.. go back
                 ****
                 ************************************************************
                 *        Subroutine to try to open in fix or var 80 format *
                 ************************************************************
                 FIXVAR
627B BEE001      ST      &00000100,RAM(FLG(SREF))   Input mode.
627E 1C04
6280 B249FD      RB      @FLAG2,VAR      Fix format flag is on.
                 LOAD2
6283 BD561C      DST     @SREF,@FAC12
6286 A35600      DADD    PABLEN-1,@FAC12  Set pointer to name length
6289 09
628A 060010      CALL    DSRLNK          Open it.
628D 08          DATA    8
628E 6871        BS      ERRIO           File not found
6290 DAE001      CLOG    >E0,RAM(FLG(SREF))   Check for error code.
6293 1CE0
6295 429C        BR      TRY
6297 BEB01C      ST      02,RAM(@SREF)   Output mode
629A 02
629B 00          RTN                     No error, go back to the caller
629C BC00E0      TRY     ST      RAM(FLG(SREF)),@TEMP   Copy flag byte
629F 011C
```

```
62A1 B2001F  363          AND    >1F,@TEMP       Delete the error code
62A4 D60004  364          $IF    @TEMP .EQ. 4 THEN                          1
62A7 42B3
62A9 BEE001  365          ST     &00010100,RAM(FLG(SREF)) Try with variable rel
62AC 1C14
62AE B64902  366          SB     @FLAG2,VAR      Variable 80 format now.    1
62B1 4283    367          BR     LOAD2                                      1
             368          $END   IF
62B3 4871    369          BR     ERRIO           Otherwise error!!!
             370  ******
             371  ***************************************************************
             372  *        SAVE SOURCE FILE                                     *
             373  ***************************************************************
             374  ESAVE
62B5 BEE001  375          ST     02,RAM(FLG(SREF))    Output mode
62B8 1C02
62BA DA4902  376          $IF    .BIT(VAR) @FLAG2 .EQ. 1 THEN               1
62BD 62C4
62BF BEE001  377          ST     >12,RAM(FLG(SREF))   Var 80 output mode    1
62C2 1C12
             378          $END   IF
62C4 0663E9  379          CALL   DSR3                 Open it.
             380  *
62C7 BEB01C  381          ST     03,RAM(@SREF)   Store OP code for write
62CA 03
62CB BEE005  382          ST     80,RAM(CHRCNT(SREF))    Store character count.
62CE 1C50
             383  *
             384  *   Write one record at a time.
             385  *
             386  WRITE
6200 0F22    387          XML    SAVEX           Execute SAVE in editor.
62D2 67C7    388          BS     EMSG            SAVE error
62D4 0663E5  389          CALL   CLOSE           Close the file
62D7 4133    390          BR     EDIT            Go back to the editor screen
             391  *
             392  *
             393  ***************************************************************
             394  *        PRINT ROUTINE TO PRINT OUT FILES                     *
             395  ***************************************************************
             396  PRINT
62D9 06627B  397          CALL   FIXVAR          Open with fix or var 80 format
             398  *        ST     02,RAM(@SREF)   Ready for read!
             399  *
             400  *        Input file is defined. Now get output device
             401  *
62DC 08FF02  403          FMT    XPT=2,YPT=21,':DEVICE NAME?:'
62EF BF2002  405          DST    >2C2,@LOC
62F2 C2
62F3 BF1C11  406          DST    PAB2,@SREF      Open next file
62F6 00
62F7 0663A0  407          CALL   DSR1            Get name and prepare PAB
62FA BEE005  408          ST     80,RAM(CHRCNT(SREF))    Store character count
62FD 1C50
62FF D7B04E  409          $IF    RAM(@FAC4) .DEQ. :TP: THEN    Check which TP
```

```
6302 545043
6305 0A
6306 86E004  410          CLR    RAM(LEN(SREF))  Change logical record length  1
6309 1C
             411          $END   IF
630A BEE001  412          ST     >12,RAM(FLG(SREF))  Open as variable 80 display
630D 1C12
630F BFE002  413          DST    BUF1,RAM(BUF(SREF))  Force to use same buffer a
6312 1C1080
6315 0663E9  414          CALL   DSR3            Open it!
6318 D6E004  415          $IF    RAM(LEN(SREF)) .EQ. >20 THEN   Old TP!!!!      1
631B 1C2043
631E 22
631F B64802  416          SB     @FLAG,TP        Set a flag                     1
             417          $END   IF
6322 BEB01C  418          ST     03,RAM(@SREF)   Ready for print!
6325 03
             419  *
6326 DA4802  420          $IF    .BIT(TP) @FLAG .EQ. 0 GOTO FAST   RS232 case
6329 638D
             421  PRINT1
632B BF1C10  422          DST    PAB1,@SREF      Now get first file
632E 00
632F 0663D0  423          CALL   CLRBUF          Clean up the buffer area
6332 0663E9  424          CALL   DSR3            Read one record
6335 BF1C11  425          DST    PAB2,@SREF      Prepare for print !!!!
6338 00
             426  *
             427  *       Next routine checks the buffer and replaces characters
             428  *
6339 BF0010  429          DST    BUF1,@TEMP          Check buffer area for CC
633C 80
633D A30000  430          DADD   79,@TEMP            Check from end of record
6340 4F
6341 BE0250  431          ST     80,@TEMP2           Set the counter
             432  LOOP
6344 BC03B0  433          ST     RAM(@TEMP),@TEMP3   Get character code.
6347 00
6348 D60320  434          $IF    @TEMP3 .EQ. SPACE GOTO MORE   Skip space
634B 6361
634D D6030C  435          $IF    @TEMP3 .EQ. >0C THEN  If page feed skip lines 1
6350 435F
6352 BEB000  436          ST     SPACE,RAM(TEMP)   Put space in.               1
6355 20
6356 0663E9  437          CALL   DSR3            Just skip a few lines         1
6359 0663E9  438          CALL   DSR3                                          1
635C 0663E9  439          CALL   DSR3                                          1
             440          $END   IF
635F 4367    441          BR     PRINT2          No need to check. Print out.
             442  MORE
6361 930D    443          DDEC   @TEMP           Decrement the buffer pointer
6363 9202    444          DEC    @TEMP2          Decrement counter
6365 6344    445          BR     LOOP            Not passed the last loc.
             446  **********
             447  *       DST    PAB2,@SREF      Already specified.
```

```
                PRINT2
6367 0663E9             CALL    DSR3            Call to write one line
                *
                *   The following section takes care of Thermal Printer case
                *
636A A60220             SUB     >20,@TEMP2          Find how many more to move
636D CE0200             $IF     @TEMP2 .GT. 0 THEN  More bytes to print?     1
6370 4380
6372 A3E002             DADD    >20,RAM(BUF(SREF))  Shift buffer address     1
6375 1C0020
6378 BCE005             ST      @TEMP2,RAM(CHRCNT(SREF)   Put count          1
637B 1C02
637D 056367             B       PRINT2              Print one more line      1
                        $END    IF
6380 BFE002             DST     BUF1,RAM(BUF(SREF))  Readjust pointer
6383 1C1080
6386 BEE005             ST      >20,RAM(CHRCNT(SREF))  Readjust character count
6389 1C20
638B 432B               BR      PRINT1              Print more line
                *****
                ****
                FAST
638D BF1C10             DST     PAB1,@SREF          Read a record
6390 00
6391 0663D0             CALL    CLRBUF              Clean up buffer
6394 0663E9             CALL    DSR3
6397 BF1C11             DST     PAB2,@SREF          Print it out
639A 00
639B 0663E9             CALL    DSR3
639E 438D               BR      FAST                Repeat the process
                ***************************************************************
                *       DSR PREPARATION AND CALL ROUTINES                     *
                ***************************************************************
                DSR1
63A0 0663D0             CALL    CLRBUF            Clean up the PAB buffer area
                DSR2
63A3 0663FE             CALL    CLRPAB            Clean up PAB area
63A6 BDE002             DST     @SREF,RAM(BUF(SREF))    Find buffer address
63A9 1C1C
63AB A3E002             DADD    >80,RAM(BUF(SREF))      Buffer address
63AE 1C0080
63B1 BEE008             ST      SCREEN,RAM(SCR(SREF))   Screen offset
63B4 1C00
63B6 BFE004             DST     >5000,RAM(LEN(SREF))    Record length
63B9 1C5000
63BC 06640A             CALL    NAME                    Get the name
63BF 8F5063             $IF     @FAC6 .DNE. 0 THEN      Do not move 0 bytes   1
63C2 CF
                COPYNM
63C3 3450E0             MOVE    @FAC6 FROM RAM(@FAC4) TO RAM(PABLEN(SREF)) Nam1
63C6 0A1CB0
63C9 4E
63CA BCE009             ST      @FAC6+1,RAM(NLEN(SREF))  Put name length in PAB1
63CD 1C51
                        $END    IF
```

```
63CF 00        488            RTN
               489   ****
               490   ****
               491   CLRBUF
63D0 A31C00    492            DADD    >80,@SREF         @SREF= buffer address
63D3 80
63D4 BEB01C    493            ST      SPACE,RAM(@SREF)  Clean up the PAB buffer.
63D7 20
63D8 35004F    494            MOVE    79 FROM RAM(@SREF) TO RAM(1(SREF))
63DB E0011C
63DE B01C
63E0 A71C00    495            DSUB    >80,@SREF         @SREF= PAB address
63E3 80
63E4 00        496            RTN
               497   ******
               498   ****************************************************************
               499   *        CLOSE A FILE                                          *
               500   ****************************************************************
               501   *
               502   CLOSE
63E5 BEB01C    503            ST      1,RAM(@SREF)      Store close op code.
63E8 01
               504   DSR3
63E9 BD561C    505            DST     @SREF,@FAC12
63EC A35600    506            DADD    PABLEN-1,@FAC12   Pointer to name length.
63EF 09
63F0 060010    507            CALL    DSRLNK               Call DSR routine
63F3 08        508            DATA    8
63F4 6871      509            BS      ERRI0                File not found
63F6 DAE001    510            CLOG    >E0,RAM(FLG(SREF))   Any other error?
63F9 1CE0
63FB 4871      511            BR      ERRI0
63FD 00        512            RTN
               513   *
               514   *
               515   ****
               516   CLRPAB
63FE 86B01C    517            CLR     RAM(@SREF)
6401 350045    518            MOVE    MAXLEN+9 FROM RAM(@SREF) TO RAM(1(SREF))  Clear
6404 E0011C
6407 B01C
6409 00        519            RTN
               520   ******
               521   ****     Get file name from screen
               522   *            FAC4 = points to file name in VDP
               523   *            FAC6 = 2 bytes name length
               524   *
               525   NAME
640A 06643D    526            CALL    SCAN50           Get user input.. file name
640D BD2024    527            DST     @STLN,@LOC       Relocate to scan file name.
6410 BE003C    528            ST      MAXLEN,@TEMP     Set the counter.
6413 8750      529            DCLR    @FAC6            Clear length of file name.
               530   SKIPSP
6415 D6B020    531            $IF     RAM(@LOC) .EQ. SPACE THEN     Skip spaces
6418 20442B
```

```
641B 9120    532          DINC   @LOC            Check next location.        1
641D 9200    533          DEC    @TEMP           Decrement counter.          1
641F 4415    534          BR     SKIPSP          Not finished scanning.      1
6421 BD2024  535          DST    @STLN,@LOC      Input file name again.      1
6424 DA4804  536          $IF    .BIT(NOTHIN) @FLAG .EQ. 1 GOTO NAMEND   No inpu1
6427 443C
6429 440A    537          BR     NAME            No file name on the line.   1
             538          $END   IF
642B BD4E20  539          DST    @LOC,@FAC4      Starting address of file name.
             540  COUNT
642E D6B020  541          $IF    RAM(@LOC) .NE. SPACE THEN                   1
6431 20643C
6434 9150    542          DINC   @FAC6           Increment name length.      1
6436 9120    543          DINC   @LOC            Look further.               1
6438 9200    544          DEC    @TEMP           Decrement counter.          1
643A 442E    545          BR     COUNT           Check more in the line.     1
             546          $END   IF
             547  NAMEND
643C 00      548          RTN
```

```
                 550  *******
                 551  ***       SCAN ROUTINE TO INPUT FILE NAME
                 552  ******
                 553  SCAN50
643D BE181F      554            ST     CURSOR,@CHRBUF   Load cursor character
6440 BD2420      555            DST    @LOC,@STLN       Copy start address
6443 BD2620      556            DST    @LOC,@ENDLN      Copy end address
                 557  SCAN51
6446 8679        558            CLR    @TIMER           Set timer
6448 C018B0      559            EX     RAM(@LOC),@CHRBUF  Alternate chars to blink
644B 20
                 560            $REPEAT
644C 03          561            SCAN                                                1
644D 6456        562            BS     GETKEY           Scan until key is recognized 1
644F CA790F      563            $UNTIL @TIMER .HE. 15   Wait for 6/60 of a second
6452 444C
6454 4446        564            BR     SCAN51
                 565  GETKEY
6456 D6B020      566            $IF    RAM(@LOC) .EQ. CURSOR THEN  Key is detected  1
6459 1F4460
645C C01880      567            EX     RAM(@LOC),@CHRBUF     Load cursor char back  1
645F 20
                 568            $SEND  IF
6460 BD0020      569            DST    @LOC,@TEMP            Find length to the cu
6463 A50024      570            DSUB   @STLN,@TEMP
6466 CA7520      571            $IF    @KEY .L. >20 GOTO CONTO2   Control keys
6469 4485
646B DA4801      572            TBR    @FLAG,INSRT           Check for insert mode
646E 44DD        573            BR     INSO4                 Insert a character
                 574  DSP
6470 BCB020      575            ST     @KEY,RAM(@LOC)        Validate the key
6473 75
6474 C52026      576            $IF    @LOC .DH. @ENDLN THEN                       1
6477 447C
6479 BD2620      577            DST    @LOC,@ENDLN           Adjust end of line     1
                 578            $END   IF
                 579  ********
                 580  *         Cursor right
                 581  *******
                 582  RGT
647C C6013C      583            $IF    @TEMP+1 .H. >MAXLEN GOTO SCAN51  No more to th
647F 6446
6481 9120        584            DINC   @LOC             Shift cursor to the right
6483 4446        585            BR     SCAN51           Go back to scan more
                 586  ******
                 587  *         Handle control keys
                 588  ******
                 589  CONTO2
6485 B248FE      590            RB     @FLAG,INSRT      Reset insert flag
6488 D6750F      591            $IF    @KEY .EQ. BACK THEN      *** Shift-Z          1
648B 4499
648D DA4820      592            $IF    .BIT(ASRC) @FLAG .EQ. 1 GOTO ERRIO2  From assel
6490 4893
6492 DA4804      593            $IF    .BIT(NOTHIN) @FLAF .EQ. 1 GOTO START  From exel
6495 4079
```

```
6497 4133    594            BR      EDIT                                        1
             595            $END    IF
6499 D67509  596            $IF     @KEY .EQ. >09 GOTO RGT     *** Cursor right.
649C 647C
649E D67508  597            $IF     @KEY .EQ. >08 THEN         *** Cursor left.   1
64A1 44AB
64A3 8E0164  598            $IF     @TEMP+1 .EQ. 0 GOTO SCAN51   Can't go left any1
64A6 46
64A7 9320    599            DDEC    @LOC                 Move cursor to the left  1
64A9 4446    600            BR      SCAN51               Go back to scan more     1
             601            $END    IF
64AB D67500  602            $IF     @KEY .EQ. >0D GOTO SCNEND  *** Enter line
64AE 6503
64B0 D67503  603            $IF     @KEY .EQ. >03 THEN         *** Delete key     1
64B3 44D3
64B5 BEB020  604            ST      SPACE,RAM(@LOC)            Store blank        1
64B8 20
64B9 BD0026  605            DST     @ENDLN,@TEMP               Find how many bytes1
64BC A50020  606            DSUB    @LOC,@TEMP                                    1
64BF CE0100  607            $IF     @TEMP+1 .LE. 0 GOTO SCN51  Nothing to the righ1
64C2 4446
64C4 3400B0  608            MOVE    @TEMP FROM RAM(1(LOC)) TO RAM(@LOC)  Move rest1
64C7 20E001
64CA 20
64CB BEB026  609            ST      SPACE,RAM(@ENDLN)       Delete last character 1
64CE 20
64CF 9326    610            DDEC    @ENDLN                  Adjust end of line    1
64D1 4446    611            BR      SCAN51                  Go back to scan again 1
             612            $END    IF
64D3 D67504  613            $IF     @KEY .EQ. >04 THEN      *** Insert key        1
64D6 4501
64D8 B64801  614            SB      @FLAG,INSRT             Turn on flag          1
64DB 4446    615            BR      SCAN51                  Go back to scan       1
             616   INS04
64DD BD0026  617            DST     @ENDLN,@TEMP            Two lines filled?     1
64E0 A50024  618            DSUB    @STLN,@TEMP             Find length of file   1
64E3 C6013C  619            $IF     @TEMP+1 .H. MAXLEN GOTO SCAN51  No space to in1
64E6 6446
64E8 BD0026  620            DST     @ENDLN,@TEMP         Find bytes to move.      1
64EB A50020  621            DSUB    @LOC,@TEMP                                    1
64EE 9100    622            DINC    @TEMP                Number of bytes to move  1
64F0 3400A3  623            MOVE    @TEMP FROM RAM(@LOC) TO RAM(>3C0)  Move to tem1
64F3 C0B020
64F6 3400E0  624            MOVE    @TEMP FROM RAM(>3C0) TO RAM(1(LOC))  Move back1
64F9 0120A3
64FC C0
64FD 9126    625            DINC    @ENDLN                  Adjust end of line    1
64FF 4470    626            BR      DSP                     Activate the character1
             627            $END    IF
6501 4446    628            BR      SCAN51                  Ignore other control
             629   SCNEND
6503 00      630            RTN                             Finish entering name.
             631   ***
```

```
                 ******
                 ******
                 ***
                 CHKRAM
6504 BC008F                ST      @RAMTOP,@TEMP    Save data first
6507 9D00
6509 BE8F9D                ST      >FF,@RAMTOP      Try to write to ERAM to see it
650C 00FF
650E D68F9D                $IF     @RAMTOP .NE. >FF GOTO ERRRAM
6511 00FF48
6514 31
6515 868F9D                CLR     @RAMTOP          Try again with 00 to make sure.
6518 00
6519 8E8F9D                $IF     @RAMTOP .NE. 00 GOTO ERRRAM
651C 004831
651F BC8F9D                ST      @TEMP,@RAMTOP    Save it back
6522 0000
6524 00                    RTN
                 *         Load UTILITY routines to ERAM.
                 ****
                 COLOR
6525 BEA300                ST      >D0,RAM(>300)    Stop sprites
6528 D0
6529 BEA380                ST      >F5,RAM(>380)    LOAD COLOR TABLE
652C F5
652D 35001F                MOVE    31 FROM RAM(>380) TO RAM(>381)
6530 A381A3
6533 80
6534 0720                  ALL     SPACE
6536 04F5                  BACK    >F5
6538 00                    RTN
                 ****
                 COLOR2
6539 BEA380                ST      >13,RAM(>380)    Change color to green
653C 13
653D 35001F                MOVE    31 FROM RAM(>380) TO RAM(>381)
6540 A381A3
6543 80
6544 0720                  ALL     SPACE
6546 04F3                  BACK    >F3
6548 00                    RTN
                 ****
                 CLRTOP
6549 868F78                CLR     @EDTTOP     Clear 6 bytes of top memory in ERAM
654C F0
654D 350005                MOVE    5 FROM @EDTTOP TO @EDTTOP+1
6550 8F78F1
6553 8F78F0
6556 00                    RTN
```

```
                 ****
                 ****************************************************************
                 *         RUN OTHER PROGRAMS FROM DISKETTE                     *
                 ****************************************************************
                 ***
                 OTHER
6557 0720        ALL     SPACE
6559 08FF02      FMT     XPT=2,YP=2,':* RUN PROGRAM FILE *:'
655C FE0213
655F 2A2052
6562 554E20
6565 50524F
6568 475241
656B 4D2046
656E 494C45
6571 202AFB
6574 B64908      SB      @FLAG2,OTHR      Set bit for "other" program
                 *
6577 BF2001      DST     >102,@LOC        Put file name? on screen.
657A 02
657B 31000A      MOVE    10 FROM ROM(#FLN) TO RAM(@LOC)
657E B02062
6581 5B
6582 A32000      DADD    >40,@LOC
6585 40
6586 B64804      SB      @FLAG,NOTHIN
6589 06640A      CALL    NAME             Get file name input.
                 *
658C 8F5045      $IF     @FAC6 .DEQ. 0 THEN   Default file name UTIL   1
658F 97
6590 0665B2      CALL    DOWNLD           Call down load program       1
6593 662D        DATA    #UTNAME          Using DSK1.UTIL1             1
6595 45A6        $ELSE                                                 1
                 *
6597 0665BF      CALL    DOWN02           Prepare PAB area             1
659A 0663C3      CALL    COPYNM           Copy name into PAB area      1
659D D7B04E      $IF     RAM(@FAC4) .DEQ. :CS: THEN  Cassette INTERFACE2
65A0 435345
65A3 A6
65A4 0720        ALL     SPACE            Clear screen for prompt      2
                 $END    IF
                 $END    IF
65A6 0665D4      CALL    DOWN10           Load it.
                 *   Following branches to >8300(@TEMP) to execute the program
65A9 066539      CALL    COLOR2      Color for execution time
65AC 0FF0        XML     >F0         Get the first address and branch
65AE 67F7        BS      ERRXML      In case of error
LAST ERROR FOUND AT LINE   224
***** ERROR IN IMMEDIATE FIELD  LINE=  698  COLUMN= 17
***** ERROR IN BRANCH ADDRESS  LINE=  698
65B0 0000        BR      EX20        End of execution code
                 ***
                 ****************************************************************
                 *         GENERAL DOWN LOAD PROGRAM                            *
                 ****************************************************************
                 *         First 6 bytes of the program file consists of :      *
```

```
                704  *         1st word:  0000 if no more file to load.            *
                705  *                     Non-zero if there is more to load        *
                706  *         2nd word:  length of the program file               *
                707  *         3rd word:  RAM address to load the program          *
                708  *   These information will be stored in CPU at >8310(@DSR)    *
                709  ***************************************************************
                710  *
                711  DOWNLD
65B2 8810       712            FETCH   @DSR            Get the file name addr
65B4 8811       713            FETCH   @DSR+1
65B6 330005     714            MOVE    5 FROM ROM(@DSR) TO RAM(PAB+15)   Copy name
65B9 AF100F
65BC 000010
                715  DOWN02
65BF 8700       716            DCLR    @TEMP           Used as XML branch address
65C1 BF1C10     717            DST     PAB1,@SREF      Get PAB addr
65C4 00
65C5 31000F     718            MOVE    15 FROM ROM(#LDDATA) TO RAM(@SREF)  Prepare PAB
65C8 B01C66
65CB 14
65CC 310014     719            MOVE    20 FROM ROM(#LODMSG) TO RAM(>202)  Display mess
65CF A20267
65D2 8B
65D3 00         720            RTN
                721  DOWN10
65D4 0663E9     722            CALL    DSR3            Load the program
65D7 350006     723            MOVE    6 FROM RAM(#BUF4) TO @DSR   Copy 6 bytes in
65DA 10AF13
65DD 80
65DE 8F0045     724            $IF     @TEMP .DEQ. 0 THEN  Copy address of first execl
65E1 E5
65E2 BD0014     725            DST     @DSR+4,@TEMP                                  1
                726            $END    IF
65E5 A71483     727            DSUB    OFFSET,@DSR+4   Take away offset
65E8 00
65E9 3412C0     728            MOVE    @DSR+2 FROM RAM(#BUF4+6) TO @0(DSR+4)  Move to
65EC 0014AF
65EF 1386
65F1 8F1066     729            $IF     @DSR .DNE. 0 THEN   If there are more files tol
65F4 05
65F5 8702       730            DCLR    @TEMP2          TEMP2 = points to last char   1
65F7 BC03AF     731            ST      RAM(PAB1+9),@TEMP2+1  Get the name length firsl
65FA 1009
65FC A30210     732            DADD    PAB1+9,@TEMP2     Add PAB address             1
65FF 09
6600 90B002     733            INC     RAM(@TEMP2)       Change last char in name    1
6603 45D4       734            BR      DOWN10            Load more file              1
                735            $END    IF
6605 BEA202     736            ST      SPACE,RAM(>202)   Clean up the screen
6608 20
6609 350013     737            MOVE    19 FROM RAM(>202) TO RAM(>203)
660C A203A2
660F 02
6610 0663FE     738            CALL    CLRPAB
6613 00         739            RTN
```

```
6614 050013  740  LDDATA   DATA   05,0,#BUF4,#0,#>2100,#10,:DSK1.:
6617 800000
661A 210000
661D 0A4453
6620 4B312E
6623 454449  741  EDNAME   DATA   :EDIT1:
6626 5431
6628 415353  742  ASNAME   DATA   :ASSM1:
662B 4D31
662D 555449  743  UTNAME   DATA   :UTIL1:
6630 4C31
```

```
                  ****** TITLE    ASSEM
                  ****************************************************************
                  *         JULY 17, 1981      Author S. Endo                    *
                  ****************************************************************
                  *         ASSEMBLER                                            *
                  ****************************************************************
                  *****
     9DD2         OPTLN     EQU     >20D2-OFFSET     Option line for assembler
                  ******
                  ASSEM
6632 0720         ALL     SPACE
6634 08FF02       FMT     XPT=2,YPT=2,':* ASSEMBLER *:'
                  *
6648 8748         DCLR    @FLAG            Clear all flags.
664A BE737E       ST      >7E,@SUBSTK      Reset subroutine stack.
664D B64820       SB      @FLAG,ASRC       Now we are in assembler
                  *
6650 D78F9D       $IF     @IDCODE .DEQ. >AA55 GOTO ASM10  It's already l
6653 00AA55
6656 6687
6658 0892A0       FMT     19<,1^,':LOAD ASSEMBLER(Y/N)? :'
                  *
6672 06679F       CALL    YORN             Get response, yes or no.
6675 06750F       $IF     @KEY .EQ. BACK GOTO START   Back key or :n: key
6678 6079
667A D67554E      $IF     @KEY .EQ. :N: GOTO START  Will take you back to
667D 6079
                  *****
                  *         LOAD ASSEMBLER HERE
                  *****
667F 0665B2       CALL    DOWNLD           Down load assembler
6682 6628         DATA    #ASNAME
6684 0665D4       CALL    DOWN10           Down load it.
                  ************
                  *  Give additional space for assembler -- COPY needs this
                  ************
                  ASM10
6687 BFAF13       DST     >0116,RAM(BUF4)
668A 800116
668D BF5613       DST     BUF4,@FAC12
6690 80
6691 BE4C04       ST      4,@FAC2        Number of files disk DSR handles
6694 060010       CALL    DSRLNK         Call files
6697 0A           DATA    10
                  ************
                  *  START OPENING FILES HERE
                  ************        Open a source file    ************
6698 08FF02       FMT     XPT=2,YPT=4,':SOURCE FILE NAME?     :'
66B5 BF1C10       DST     #PAB1,@SREF      Put source PAB address.
66B8 00
66B9 BF2000       DST     >0C2,@LOC        Get loc for source file name.
66BC C2
66BD 0663A3       CALL    DSR2             Prepare PAB area
                  *
66C0 BFE002       DST     BUF1,RAM(BUF(SREF))   Source buffer address
```

```
66C3 1C1080
             798  *         Next subroutine tries to open with fix and var format
             799  *         ST      04,RAM(FLG(SREF))   Disp, fixed, seq, input
             800  *         ST      >14,RAM(FLG(SREF))  Disp, variable, seq, input
66C6 06627B  801            CALL    FIXVAR          Open it!!!
             802  *
             803  ************   Open an object file     ************
66C9 08FF02  805            FMT     XPT=2,YPT=8,':OBJECT FILE NAME?:'
66E1 BF1C11  807            DST     #PAB2,@SREF     Put obj PAB address
66E4 00
66E5 BF2001  808            DST     >142,@LOC       Set file name pointer.
66E8 42
66E9 0663A3  809            CALL    DSR2            Prepare PAB area
             810  *
66EC BEE001  811            ST      00,RAM(FLG(SREF))     Open with update mode
66EF 1C00
66F1 BFE002  812            DST     BUF2,RAM(BUF(SREF))   Buffer address.
66F4 1C1180
66F7 0663E9  813            CALL    DSR3                  Open output file.
66FA B64840  814            SB      @FLAG,AOBJ            Flag for object file open
             815  *
             816  ************      Open a list file     ************
66FD 08FF02  818            FMT     XPT=2,YPT=12,':LIST FILE NAME?:'
6713 BF1C12  820            DST     #PAB3,@SREF           Put list PAB address
6716 00
6717 B64804  821            SB      @FLAG,NOTHIN          No file name is OK.
671A BF2001  822            DST     >1C2,@LOC             Set pointer to name
671D C2
671E 0663A3  823            CALL    DSR2                  Prepare PAB area.
             824  *
6721 8F5067  825            $IF     @FAC6 .DEQ. 0 GOTO ASKOPT   No list file. Skip.
6724 36
6725 BEE001  826            ST      >12,RAM(FLG(SREF))    Open with output mode
6728 1C12
672A BFE002  827            DST     BUF3,RAM(BUF(SREF))   Buffer address.
672D 1C1280
6730 0663E9  828            CALL    DSR3                  Open output file
6733 B64880  829            SB      @FLAG,ALST            List file is open
             830  *
             831  ************   Ask for options     ************
             832  ASKOPT
             833  *  Clean up OPTLN beforehand ??????
6736 08FF02  834            FMT     XPT=2,YPT=16,':OPTIONS?:'
6739 FE1007
673C 4F5054
673F 494F4E
6742 533FFB
6745 BF2002  835            DST     >242,@LOC         Specify cursor location
6748 42
6749 06640A  836            CALL    NAME
674C B248FB  837            RB      @FLAG,NOTHIN      No input line was OK so far.
             838  *  Option is transferred to ERAM
674F 8F5047  839            $IF     @FAC6 .DEQ. 0 THEN    No option. It's OK      1
6752 56
6753 BD4E20  840            DST     @LOC,@FAC4                                    1
```

```
                841            $END    IF
6756 35000F     842            MOVE    15 FROM RAM(@FAC4) TO @OPTLN   Move to RAM
6759 8F9DD2
675C B04E
                843   ***
                844   ************   Now assemble !!!!!!  *********
                845   ***
675E 066549     846            CALL    CLRTOP         Clear editor information
6761 0720       847            ALL     SPACE
6763 0F21       848            XML     ASM            Assemble the progam.
6765 67C7       849            BS      EMSG           Error routine shared with editor
                850   *********   Close files before going back to menu   ******
                851   *
                852   ASMOUT
6767 D66549     853            CALL    CLRTOP         Clear editor information
676A BF1C10     854            DST     PAB1,@SREF     Close source file
676D 00
676E 0663E5     855            CALL    CLOSE
6771 BF1C11     856            DST     PAB2,@SREF     Close object file
6774 00
6775 0663E5     857            CALL    CLOSE
6778 DA4880     858            $IF     .BIT(ALST) @FLAG .EQ. 1 THEN                  1
677B 6784
677D BF1C12     859            DST     PAB3,@SREF     Close list file                1
6780 00
6781 0663E5     860            CALL    CLOSE                                         1
                861            $END
6784 8648       862            CLR     @FLAG
                863   *********
6786 06693B     864            CALL    CONT           Issue message and continue.
6789 4079       865            BR      START          Go back to menu screen.
                866   *******
                867   *******
                868   ******
                869   *        DATA AREA
                870   *****
6788 4F4E45     872   LODMSG   DATA    :ONE MOMENT PLEASE...:
```

```
                875  ******
                876  ****
                877  *****
                878  *         Get "yes" or "no" response from keyboard
                879  *****
                880  YORN
679F  BE181F    881            ST      CURSOR,@CHRBUF
                882  SCAN30
67A2  8679      883            CLR     @TIMER
67A4  C0187D    884            EX      CB,@CHRBUF
                885            $REPEAT                 Waiting for input.
67A7  03        886            SCAN                                          1
67A8  67B1      887            BS      SCAN35                                1
67AA  CA7906    888            $UNTIL  @TIMER .HE. 6
67AD  47A7
67AF  47A2      889            BR      SCAN30
                890  SCAN35
67B1  D6750F    891            $IF     @KEY .EQ. BACK GOTO YNEND
67B4  67C0
67B6  D67559    892            $IF     @KEY .EQ. :Y: GOTO YNEND
67B9  67C0
67BB  D6754E    893            $IF     @KEY .NE. :N: GOTO SCAN30
67BE  47A2
                894  YNEND
67C0  BC7D75    895            ST      @KEY,CB         Show character on screen.
67C3  00        896            RTN
                897  ****
                898  ***
                899  *
```

```
                901  ***       TITLE   ERROR
                902  ***************************************************************
                903  *         July 17, 1981      Author S. Endo                   *
                904  ***************************************************************
                905  *         ERROR HANDLING ROUTINES                             *
                906  ***************************************************************
                907  *                                                             *
                908  *   All GPL errors are handled in this section                *
                909  ***************************************************************
                910  *   EMSG -- error message returned from assembly language     *
                911  *           utility program such as ASSM, EDIT, ALUTIL        *
                912  *******
                913  EMSG1
67C4 066525     914            CALL    COLOR       Special entry for editor
                915  EMSG
67C7 066810     916            CALL    FIX         Fix screen and branch BASIC error
67CA CA2208     917            $IF     @ERRCOD .L. 8 GOTO ERRIO
67CD 4871
67CF A62208     918            SUB     8,@ERRCOD
67D2 C62205     919            $IF     @ERRCOD .H. 5 GOTO RETURN   No error
67D5 67E8
67D7 0668A7     920            CALL    CLSALL      Close all files. No error checks.
67DA 8A22       921            CASE    @ERRCOD
67DC 4823       922            BR      ERRMEM      Code 8: Memory full
67DE 481C       923            BR      WRNCHR      Code 9: Control character
67E0 483C       924            BR      BADTAG      Code A: Bad tag
67E2 4843       925            BR      ERRCSM      Code B: Checksum error
67E4 484A       926            BR      ERRDDF      Code C: Duplicate definition
67E6 4851       927            BR      ERRURR      Code D: Unresolved references
                928  *
                929  RETURN
67E8 DA4820     930            $IF     .BIT(ASRC) @FLAG .EQ. 1 GOTO ASMOUT
67EB 4767
LAST ERROR FOUND AT LINE   698
***** ERROR IN IMMEDIATE FIELD  LINE=  931  COLUMN= 44
***** ERROR IN BRANCH ADDRESS  LINE=  931
67ED DA4901     931            $IF     .BIT(LD) @FLAG2 .EQ. 1 GOTO LOAD
67F0 6067
67F2 0668C4     932            CALL    CLSNO
67F5 4133       933            BR      EDIT            Go back to editor
                934  ***************************************************************
                935  *         ERRXML -- Error returned from user's program        *
                936  *                    or program file loaded by selection S    *
                937  ***************************************************************
                938  ERRXML
67F7 066810     939            CALL    FIX        Fix screen and branch to BASIC error
67FA 066525     940            CALL    COLOR      Reload screen
                941  *         The following error can only be used by user execution
67FD D6220F     942            $IF     @ERRCOD .EQ. >0F THEN                        1
6800 4809
6802 0668E4     943  ERRPR2    CALL    ERROR      :Program not found:               1
6805 6A29       944            DATA    #PRGNF                                       1
6807 485D       945            BR      ERRNM2     Clear msg. Go back for new input.  1
                946            $END    IF
6809 0668E4     947            CALL    ERROR      :XML error:
680C 69BF       948            DATA    #XMLERR
```

```
680E 4029    949            BR      BEGIN2    Initialize all except top memory
             950   ***
             951   *        FIX -- Fix screen and branch to BASIC errors
             952   ****
             953   FIX
6810 390001  954            MOVE 1 FROM ROM(#ON) TO VDP(1)   Turn on the screen
6813 016271
6816 DA4808  955            $IF     .BIT(BSC) @FLAG .EQ. 1 GOTO ERRBSC   Error in BA
6819 4D9C
681B 00      956            RTN
             957   **
             958   ***************************************************************
             959   *        General warning routines                             *
             960   ***************************************************************
             961   WRNCHR
681C 066808  962            CALL    WARN      :Control character removed:
681F 69CA    963            DATA    #CHRWRN
6821 47E8    964            BR      RETURN
             965   *
             966   ***************************************************************
             967   *        General error routines                               *
             968   ***************************************************************
             969   ERRMEM
6823 0668E4  970            CALL    ERROR     :memory full error:
6826 6982    971            DATA    #MEM
6828 47E8    972            BR      RETURN
             973   *
             974   ERRPRG
682A 0668E4  975            CALL    ERROR     :program not found:
682D 6A29    976            DATA    #PRGNF
682F 4079    977            BR      START     Go back to menu screen
             978   *                           Nothing is destroyed yet.
             979   *
             980   ERRRAM
6831 DA4808  981            $IF     .BIT(BSC) @FLAG .EQ.1 GOTO ERRRMB  Go back to B
6834 4E57
6836 0668E4  982            CALL    ERROR     :no memory expansion:
6839 699D    983            DATA    #RAMERR
683B 0B      984            EXIT
             985   *
             986   BADTAG
683C 0668E4  987            CALL    ERROR     :ILLEGAL TAG:
683F 69E4    988            DATA    #ILLTAG
LAST ERROR FOUND AT LINE   931
***** ERROR IN IMMEDIATE FIELD  LINE=  989  COLUMN= 17
***** ERROR IN BRANCH ADDRESS  LINE=  989
6841 0000    989            BR      LOAD
             990   *
             991   ERRCSM
6843 0668E4  992            CALL    ERROR     :checksum error:
6846 69F0    993            DATA    #CSMERR
LAST ERROR FOUND AT LINE   989
***** ERROR IN IMMEDIATE FIELD  LINE=  994  COLUMN= 17
***** ERROR IN BRANCH ADDRESS  LINE=  994
6848 0000    994            BR      LOAD
             995   *
```

```
684A 0668E4  997          CALL   ERROR     :duplicate definition:
684D 69FF    998          DATA   #DUPDEF
LAST ERROR FOUND AT LINE   994
***** ERROR IN IMMEDIATE FIELD  LINE=  999  COLUMN= 17
***** ERROR IN BRANCH ADDRESS  LINE=  999
684F 0000    999          BR     LOAD
             1000  *
             1001  ERRURR
6851 0668E4 1002          CALL   ERROR     :unresolved references:
6854 6A14   1003          DATA   #UNRREF
LAST ERROR FOUND AT LINE   999
***** ERROR IN IMMEDIATE FIELD  LINE= 1004  COLUMN= 17
***** ERROR IN BRANCH ADDRESS  LINE= 1004
6856 0000   1004          BR     LOAD
             1005  *
             1006  ERRNAM
6858 0668E4 1007          CALL   ERROR     :name too long:
685B 69B1   1008          DATA   #NAM
             1009  ERRNM2
685D 8F5068 1010          $IF    @FAC6 .DEQ. 0 GOTO ERRNM3
6860 6C
6861 BEB04E 1011          ST     SPACE,RAM(@FAC4)   Delete name on screen
6864 20
6865 3450E0 1012          MOVE   @FAC6 FROM RAM(@FAC4) TO RAM(1(FAC4))
6868 014EB0
686B 4E
             1013  ERRNM3
686C 06694B 1014          CALL   CLNMSG             Clean up message area
LAST ERROR FOUND AT LINE  1004
***** ERROR IN IMMEDIATE FIELD  LINE= 1015  COLUMN= 17
***** ERROR IN BRANCH ADDRESS  LINE= 1015
686F 0000   1015          BR     EXEC
             1016  ***************************************************************
             1017  *        ERRIO -- section for IO errors                       *
             1018  ***************************************************************
             1019  ERRIO
6871 BC4EED 1020          ST     RAM(FLG(SREF)),@FAC4  Get error code out.
6874 011C
6876 B24EE0 1021          AND    >E0,@FAC4             Take error code bits
6879 E64E05 1022          SRL    @FAC4,5             Move 5 bits to right justif
687C B64E30 1023          OR     >30,@FAC4             Add ASCII offset.
             1024  *
687F B2E001 1025          AND    >1F,RAM(FLG(SREF))    Get error code out
6882 1C1F
             1026  *
             1027  *   Next line checks if error is reading after end-of-file.
             1028  *
6884 D64E35 1029          $IF    @FAC4 .EQ. >35 THEN   In case of just an end-o1
6887 488E
6889 0668A7 1030          CALL   CLSALL                Close all the files     1
688C 4133   1031          BR     EDIT                  Branch back to editor   1
             1032          $END   IF
             1033  *
688E 0668E4 1034          CALL   ERROR                 Regular I/O error.
6891 698E   1035  DATA    #IO
             1036  ERRIO2
6893 0668A7 1037          CALL   CLSALL    Close all files without checking err
             1038  ***
```

```
6896 DA4820 1039          $IF     .BIT(ASRC) @FLAG .EQ. 1 GOTO START
6899 4079
LAST ERROR FOUND AT LINE  1015
***** ERROR IN IMMEDIATE FIELD  LINE= 1040  COLUMN= 40
***** ERROR IN BRANCH ADDRESS  LINE= 1040
689B DA4901 1040          $IF     .BIT(LD) @FLAG .EQ. 1 GOTO LOAD
689E 6079
68A0 DA4908 1041          $IF     .BIT(OTHR) @FLAG .EQ. 1 GOTO BEGIN
68A3 4025
68A5 4133   1042          BR      EDIT                    Go back to editor
            1043  ****
            1044  CLSALL
68A7 BF1C10 1045          DST     PAB1,@SREF              Close first file
68AA 00
68AB 0668C4 1046          CALL    CLSNO
68AE DF1C11 1047          DST     PAB2,@SREF              Close second file
68B1 00
68B2 0668C4 1048          CALL    CLSNO
68B5 BF1C12 1049          DST     PAB3,@SREF              Close third file
68B8 00
68B9 0668C4 1050          CALL    CLSNO
68BC BF1C13 1051          DST     PAB4,@SREF  Close fourth file used in COPY stat
68BF 00
68C0 0668C4 1052          CALL    CLSNO
68C3 00     1053          RTN
            1054  ****
            1055  CLSNO
68C4 BD561C 1056          DST     @SREF,@FAC12            Pointer to name length
68C7 A35600 1057          DADD    PABLEN-1,@FAC12
68CA 09
68CB BEB01C 1058          ST      1,RAM(@SREF)            Close file
68CE 01
68CF 060010 1059          CALL    DSRLNK          Call DSR -- no worry about erro
68D2 08     1060          DATA    8
68D3 86E009 1061          CLR     RAM(NLEN(SREF))  Clear name len to avoid prob wi
68D6 1C
68D7 00     1062          RTN
            1063  ****
            1064  ***************************************************************
            1065  *         WARNING HANDLING SECTION                            *
            1066  ***************************************************************
            1067  WARN
68D8 06694B 1068          CALL    CLNMSG                  Clean up message area
68DB 31000B 1069          MOVE    11 FROM ROM(#WRNM) TO RAM(>2A2)
68DE A2A269
68E1 77
            1070          BR      WRNENT
            1071  *
            1072  ***************************************************************
            1073  *         ERROR HANDLING SECTION                              *
            1074  ***************************************************************
            1075  ERROR
68E4 06694B 1076          CALL    CLNMSG                  Cleanup message area
68E7 310009 1077          MOVE    9 FROM ROM(#ERRM) TO RAM(>2A2)
68EA A2A269
68ED 57
```

```
            1078  WRNENT
68EE 884A   1079          FETCH   @FAC                    FETCH MESSAGE ADDRESS.
68F0 884B   1080          FETCH   @FAC+1
68F2 864C   1081          CLR     @FAC+2
68F4 330001 1082          MOVE    1 FROM ROM(@FAC) TO @FAC+3
68F7 4D0000
68FA 4A
68FB 324CA2 1083          MOVE    @FAC+2 FROM ROM(1(FAC)) TO RAM(>2C2)
68FE C20001
6901 4A
6902 D74A69 1084          $IF     @FAC .DEQ. #IO THEN    If an I/O error         1
6905 8E490C
6908 BCA2D2 1085          ST      @FAC4,RAM(>2D2)        Display error code      1
690B 4E
            1086          $END    IF
690C D74A69 1087          $IF     @FAC .DEQ. #XMLERR THEN                        1
690F BF493B
6912 E72200 1088          DSRL    @ERRCOD,4              Change error code to    1
5915 04
6916 E62304 1089          SRL     @ERRCOD+1,4                                    1
6919 C62209 1090          $IF     @ERRCOD .H. 9 THEN     Take care of A - F      2
691C 4926
691E C6220F 1091          $IF     @ERRCOD .H. 15 GOTO CONT Number too big so st 2
6921 693B
6923 A22207 1092          ADD     7,@ERRCOD                                      2
            1093          $END    IF
6926 C62309 1094          $IF     @ERRCOD+1 .H. 9 THEN   Take care of A - F      2
6929 4933
692B C6230F 1095          $IF     @ERRCOD+1 .H. 15 GOTO CONT   Skip this number 2
692E 693B
6930 A22307 1096          ADD     7,@ERRCOD+1                                    2
            1097          $END    IF
6933 A32230 1098          DADD    >3030,@ERRCOD                                  1
6936 30
6937 BDA2D0 1099          DST     @ERRCOD,RAM(>2D0)      Display it on screen    1
693A 22
            1100          $END    IF
            1101  CONT
693B 310017 1102          MOVE    23 FROM ROM(#PRESS) TO RAM(>2E2)
693E A2E269
6941 60
            1103  SCAN40
6942 03     1104          SCAN
6943 4942   1105          BR      SCAN40                 Wait till hit enter.
6945 D6750D 1106          $IF     @KEY .NE. ETR GOTO SCAN40
6948 4942
694A 00     1107          RTN
            1108  ****
            1109  CLNMSG
694B BEA2A0 1110          ST      SPACE,RAM(>2A0)        CLEAN UP MSG AREA.
694E 20
694F 35005F 1111          MOVE    >5F FROM RAM(>2A0) TO RAM(>2A1)
6952 A2A1A2
6955 A0
6956 00     1112          RTN
```

```
            1113  *
            1114  ********
            1115  *         DATA AREA
            1116  *******
6957 2A2045 1118  ERRM      DATA    :* ERROR *:
6960 505245 1119  PRESS     DATA    :PRESS ENTER TO CONTINUE:
6977 2A2057 1120  WRNM      DATA    :* WARNING *:
6982 0B4D45 1121  MEM       DATA    11,:MEMORY FULL:
698E 0E492F 1122  IO        DATA    14,:I/O ERROR CODE:
699D 134E4F 1123  RAMERR    DATA    19,:NO MEMORY EXPANSION:
69B1 0D4E41 1124  NAM       DATA    13,:NAME TOO LONG:
69BF 0A4552 1125  XMLERR    DATA    10,:ERROR CODE:
69CA 19434F 1126  CHRWRN    DATA    25,:CONTROL CHAACTER REMOVED:
69E4 0B494C 1127  ILLTAG    DATA    11,:ILLEGAL TAG:
69F0 0E4348 1128  CSMERR    DATA    14,:CHECKSUM ERROR:
69FF 144455 1129  DUPDEF    DATA    20,:DUPLICATE DEFINITION:
6A14 14554E 1130  UMRREF    DATA    20,:UNRESOLVED REFERENCE:
6A29 115052 1131  PRGNF     DATA    17,:PROGRAM NOT FOUND:
            1132  ******
6A3B 707070 1133  CDATA     DATA    >70,>70,>70,>70,>70,>70,>70,>70
6A43 007E42 1134            DATA    00,>7E,>42,>42,>42,>42,>7E,00
6A4B E0000E 1135  VDPRG2    DATA    >E0,>0,>0E,>1,06,0,>F5
```

```
                 *         TITLE   BASIC
                 ************************************************************
                 *         Sept 09, 1981      Author S. Endo                 *
                 ************************************************************
                 *         BASIC SUPPORT FUNCTIONS                           *
                 ************************************************************
                 *****
                 *         This program takes care of interface to BASIC.
                 *         Routines callable from console BASIC are listed
                 *         in the link list below.
                 *****
                 *******
                 **        XML EQUATES   ***
                 ******
0012             CFI       EQU     >12      Convert floating point to integer
0013             SYM       EQU     >13
0014             SMB       EQU     >14
0015             ASSGNV    EQU     >15      Assign value to numeric.
0017             VPUSH     EQU     >17      Push to stack
0018             VPOP      EQU     >18      Pop off the stack.
001B             PGMCH     EQU     >1B      Advance one character
                 *
                 ****      GPL CALLS     ***
                 *
0038             GETSTR    EQU     >38      GET ENOUGH SPACE FOR STRING
001A             WARN$$    EQU     >1A      Warning handling routine
001C             ERR$$     EQU     >1C      Error handling routine
                 ******
                 ***       CPU RAM EQUATES  ***
                 *****
0004             PC        EQU     >04      Address pointer
000C             BYTE      EQU     >0C      Number of bytes to move
0010             OLDS      EQU     >10
0012             CNT       EQU     >12      Counter.
0016             PCODE     EQU     >16      ERAM address to store parm information
                 *
9D0A             PARM      EQU     >200A->8300  PARAMETER COUNT INFORMATION
                 *
002C             PGMPTR    EQU     >2C      Program pointer
                 *
0040             FREPTR    EQU     >40      Free space pointer
0042             CHAT      EQU     >42      Character code
006E             VSPTR     EQU     >6E      Value stack pointer
                 ****
00B3             COMMA$    EQU     >B3      Token for ","
00B6             RPAR$     EQU     >B6      Token for ")"
00B7             LPAR$     EQU     >B7      Token for "("
0065             STRING    EQU     >65      String ID byte
                 ************************************************************
                 **   LINK LIST -- list of callable subprograms              *
                 ************************************************************
6A52 6A5B6A      LINK01    DATA    #LINK02,#INIT,4,:INIT:
6A55 960449
6A58 4E4954
6A5B 6A646A      LINK02    DATA    #LINK03,#BLOAD,4,:LOAD:
```

```
6A5E EC044C
6A61 4F4144
6A64 6A6D6C 1191  LINK03   DATA   #LINK04,#BLINK,4,:LINK:
6A67 08044C
6A6A 494E4B
6A6D 6A766B 1192  LINK04   DATA   #LINK05,#PEEK,4,:PEEK:
6A70 830450
6A73 45454B
6A76 6A806B 1193  LINK05   DATA   #LINK06,#PEEKV,5,:PEEKV:
6A79 7E0550
6A7C 45454B
6A7F 56
6A80 6A8A6A 1194  LINK06   DATA   #LINK07,#POKEV,5,:POKEV:
6A83 E70550
6A86 4F4B45
6A89 56
6A8A 00006D 1195  LINK07   DATA   #0,#GETCHR,7,:CHARPAT:
6A8D 120743
6A90 484152
6A93 504154
            1196  ***************************************************************
            1197  *   INIT -- initialization routine                            *
            1198  ***************************************************************
            1199  INIT
6A96 B64808 1200           SB     @FLAG,BSC   Set BASIC flag on.
6A99 A32C00 1201           DADD   5,@PGMPTR   Skip name len and name 'INIT'
6A9C 05
6A9D 066AAA 1202           CALL   INIT3       Force UTIL load
            1203  *
6AA0 4D01   1204           BR     RTBSC2      Go back to BASIC interpreter
            1205  ***
            1206  *   routine to load UTIL subroutines
            1207  ***
            1208  INIT2
6AA2 D78F9D 1209           $IF    @IDCODE .DNE. >A55A THEN  Check if UTIL is the1
6AA5 00A55A
6AA8 6AD1
            1210  INIT3
6AAA 066504 1211           CALL   CHKRAM      Is E-RAM there?                   1
6AAD BE4A03 1212           ST     3,@FAC      Force UTIL load                   1
6AB0 BF4C70 1213           DST    ALCODE,@FAC2                                  1
6AB3 00
     6AB4   1214  INITLP   EQU    $
6AB4 330004 1215           MOVE   4 FROM ROM(@FAC2) TO @FAC4  Load utility routi1
6AB7 4E0000
6ABA 4C
6ABB A34C00 1216           DADD   4,@FAC2                                       1
6ABE 04
6ABF 324ECF 1217           MOVE   @FAC4 FROM ROM(@FAC2) TO @->8300(FAC6)        1
6AC2 7D0050
6AC5 00004C
6AC8 A14C4E 1218           DADD   @FAC4,@FAC2                                   1
6ACB 924A   1219           DEC    @FAC                                          1
6ACD 4AB4   1220           BR     INITLP                                        1
6ACF 4549   1221           BR     CLRTOP   Clear editor information and return  1
```

```
                1222          $END    IF
6AD1 00         1223          RTN
                1224   ***
                1225   ****
                1226   SKIPN
6AD2 864A       1227          CLR     @FAC          Count size of routine name
6AD4 BC4BB0     1228          ST      RAM(@PGMPTR),@FAC+1   Copy size
6AD7 2C
6AD8 A12C4A     1229          DADD    @FAC,@PGMPTR  Advance pointer to end of name
6ADB 912C       1230          DINC    @PGMPTR       Advance pointer to char after name
6ADD 0F1B       1231          XML     PGMCH         Get next token
6ADF D642B7     1232          $IF     @CHAT .NE. LPAR$ GOTO ERRIS   Has to be "(".
6AE2 4E07
6AE4 0F1B       1233          XML     PGMCH         Advance pointer one more time
6AE6 00         1234          RTN
```

```
                  ***
                  ****************************************************************
                  *         CALL LOAD and POKEV from BASIC program        ********
                  ****************************************************************
                  POKEV
6AE7 B64801                 SB      @FLAG,VFLG    Turn on VDP flag
6AEA 4AF1                   BR      LDP1          Join LOAD routine
                  BLOAD
6AEC 8748                   DCLR    @FLAG         Delete all flags
6AEE B64808                 SB      @FLAG,BSC     Called from BASIC program.
                  *****
                  LDP1
6AF1 066AD2                 CALL    SKIPN         Skip name first
                  LDP3
6AF4 0EB6                   PARSE   RPAR$         Parse up to ")".
                  **        Process file name if it is a string.    **
6AF6 D64C65                 $IF     @FAC2 .EQ. STRING GOTO LDP2
6AF9 6B3D
                  **        Otherwise, it is an absolute address    **
6AFB 0F12                   XML     CFI           Convert to integer
6AFD D65403                 $IF     @FAC10 .EQ. 03 GOTO ERRNTS  Check for overflow
6B00 6E1B
6B02 BD044A                 DST     @FAC,@PC  Save in ERAM(or VRAM) address pointer
                  LDP4
6B05 D642B3                 $IF     @CHAT .NE. COMMA$ GOTO RTBSC   RETURN TO BASIC
6B08 4CFC
6B0A 0F1B                   XML     PGMCH         Skip ","
6B0C 0EB6                   PARSE   RPAR$         Parse up to ")"
6B0E D64C65                 $IF     @FAC2 .EQ. STRING THEN      No more data?     1
6B11 4B19
6B13 8F506B                 $IF     @FAC6 .DEQ. 0 GOTO BEXTLD   Null string?      1
6B13 75
6B17 4E25                   BR      ERRBA         Bad argument                    1
                            $END    IF
                  **        Numeric in FAC area
6B19 0F12                   XML     CFI           Convert to integer
6B1B D65403                 $IF     @FC10 .EQ. 03 GOTO ERRNTS   Overflow error
6B1E 6E1B
                  ***
6B20 DA4801                 $IF     .BIT(VFLG) @FLAG .EQ.  THEN                   1
6B23 6B2B
6B25 BCB004                 ST      @FAC+1,RAM(@PC)   POKE INTO VDP RAM LOCATION  1
6B28 4B
6B29 4B38                   $ELSE                 Poke into CPU RAM               1
6B2B A70483                 DSUB    OFFSET,@PC                                    1
6B2E 00
6B2F BCC000                 ST      @FAC+1,@0(PC)                                 1
6B32 044B
6B34 A30483                 DADD    OFFSET,@PC                                    1
6B37 00
                            $END    IF
6B38 9104                   DINC    @PC           Increment ERAM address.
6B3A 056B05                 B       LDP4          Continue loop
                  *****
                  LDP2
```

```
6B3D 8F506B 1280          $IF    @FAC6 .DEQ. 0 GOTO BEXTLD   Skip null string
6B40 75
6B41 DA4801 1281          $IF    .BIT(VFLG) .EQ. 1 GOTO ERRSNM  Not for VDP poke
6B44 4E20
            1282  *
            1283  *       Following section prepares PAB area for object file   *
            1284  *
6B46 BD0C50 1285          DST    @FAC6,@BYTE  Store actual spec. length
6B49 A30C00 1286          DADD   PABLEN+80,@BYTE  Add in the PAB len and buffer 1
6B4C 5A
6B4D 0F17   1287          XML    VPUSH        Push possible temp. string
6B4F 066AA2 1288          CALL   INIT2        make sure Utility is there
6B52 060038 1289          CALL   GETSTR       and try to allocate space
6B55 0F18   1290          XML    VPOP         Restore original string data
            1291  *
            1292  *   Variable specification
            1293  *       FAC+4,5     Start address of original device spec.
            1294  *       FAC+6,7     LENGTH OF ORIGINAL DEVICE SPECIFICATION
            1295  *       SREF        Location of PAB in VDP memory
            1296  *       BYTE        Length of entire PAB, including spec.
            1297  *
6B57 0663FE 1298          CALL   CLRPAB       Clear the entire PAB
6B5A 0663C3 1299          CALL   COPYNM       Copy device name and length
6B5D BEE008 1300          ST     >60,RAM(SCR(SREF))   Screen offset
6B60 1C60
6B62 BEE001 1301          ST     &00000100,RAM(FLG(SREF))   Disp,fixed,seq,input
6B65 1C04
6B67 A1501C 1302          DADD   @SREF,@FAC6    Calculate the address of the
6B6A A35000 1303          DADD   PABLEN,@FAC6   buffer
6B6D 0A
6B6E BDE002 1304          DST    @FAC6,RAM(BUF(SREF))  Store buffer addr. in PAB
6B71 1C50

LAST ERROR FOUND AT LINE  1040

***** ERROR IN IMMEDIATE FIELD  LINE= 1305  COLUMN= 16

***** ERROR IN BRANCH ADDRESS  LINE= 1305

6B73 0000   1305          BR     BENTLD       Enter object file loader
            1306  ****
            1307  ********                                              *******
            1308  **  Entry from object file loader  CALL LOAD              ****
            1309  ********                                              *******
     6B75   1310  BEXTLD  EQU    $
            1311  *
6B75 D642B3 1312          $IF    @CHAT .NE. COMMA$ GOTO RTBSC  Go back to BASIC
6B78 4CFC
6B7A 0F1B   1313          XML    PGMCH        {SKIP COMMA}
6B7C 4AF4   1314          BR     LDP3         {CONTINUE IN MAIN LOOP}
            1315  ********
```

```
                 ********                                                 *******
                 ************************************************************
                 *            PEEK AND PEEKV ROUTINE                          *
                 ************************************************************
                 *
                 * FORMAT:
                 *   CALL PEEK  (address, numeric variable,....)
                 *   CALL PEEKV (Address, numeric variable,....)
                 * FUNCTION:
                 *   RETURNS THE VALUE AT address IN ERAM INTO
                 *   numeric-variable.  IF MORE THAN ONE numeric-variable IS
                 *   SPECIFIED THEN address IS INCREMENTED AND THE VALUE
                 *   IN ERAM AT THE NEW address IS ASSIGNED TO THE NEXT
                 *   VARIABLE AND SO ON.
                 *
                 PEEKV
6B7E B64801      1333          SB      @FLAG,VFLG  VDP FLAG SET
6B81 4B85        1334          BR      PEEK0       ENTRY POINT
                 1335  PEEK
6B83 8748        1336          DCLR    @FLAG       CLEAR FLAG
                 1337  PEEK0
6B85 066AD2      1338          CALL    SKIPN       SKIP NAME FIRST
                 1339  * CALL INIT2     CHECK UTIL IS THERE AND LOAD
                 1340  PEEK1
6B88 0EB6        1341          PARSE   RPAR$       GET VALUE OF ADDRESS
6B8A D64C65      1342          $IF     @FAC2 .EQ. STRING GOTO PEEK3   SKIP NULL STRING
6B8D 6BAC
6B8F 0F12        1343          XML     CFI         [CONVERT FAC TO INTEGER]
6B91 D65403      1344          $IF     @FAC10 .EQ. 03 GOTO ERRNTB    [OVERFLOW]
6B94 6E1B
6B96 BD044A      1345          DST     @FAC,@PC    [SAVE PEEK ADDRESS]
6B99 D642B3      1346          $IF     @CHAT .NE. COMMA$ GOTO RTBSC  [GO BACK TO BASIC
6B9C 4CFC
                 1347  *
                 1348  PEEK2
6B9E 0F1B        1349          XML     PGMCH                 SKIP ","
6BA0 D642C7      1350          $IF     @CHAT .EQ. >C7 THEN   STRING                1
6BA3 4BB9
6BA5 0EB6        1351          PARSE   RPAR$                 GET STRING VALUE      1
6BA7 D64C65      1352          $IF     @FAC2 .EQ. STRING THEN SKIP   NULL STRING   2
6BAA 4BB9
                 1353  PEEK3
6BAC 8F504E      1354          $IF     @FAC6 .DNE. 0 GOTO ERRSNM     SHOULD BE NULL 2
6BAF 20
6BB0 D642B3      1355          $IF     @CHAT .NE. COMMA$ GOTO RTBSC  RETURN TO BASIC 2
6BB3 4CFC
6BB5 0F1B        1356          XML     PGMCH       SKIP COMMA                      2
6BB7 4B88        1357          BR      PEEK1       GO BACK TO THE LOOP             2
                 1358          $END    IF
                 1359          $END    IF
6BB9 CA4280      1360          $IF     @CHAT .HE. >80 GOTO RTBSC  GO BACK TO BASIC?
6BBC 6CFC
6BBE 0F13        1361          XML     SYM         GET SYMBOL NAME
6BC0 0F14        1362          XML     SMB         GET VALUE POINTER
6BC2 0F17        1363          XML     VPUSH       SAVE FAC AREA
```

```
            1364  *
            1365  ** GET VALUE FROM ERAM OR RAM INTO FAC+1
            1366  *
6BC4 DA4801 1367            $IF     .BIT(VFLG) @FLAG .EQ. 1 THEN                1
6BC7 6BCF
6BC9 BC5DB0 1368            ST      RAM(@PC),@ARG+1   GET VALUE FROM VDP RAM     1
6BCC 04
6BCD 4BDC   1369            $ELSE                     GET VALUE FROM CPU RAM     1
6BCF A70483 1370            DSUB    OFFSET,@PC        TAKE OUT OFFSET            1
6BD2 00
6BD3 BC5DC0 1371            ST      @0(PC),@ARG+1     MOVE VALUE FROM ERAM       1
6BD6 0004
6BD8 A30483 1372            DADD    OFFSET,@PC                                   1
6BDB 00
            1373            $END    IF
            1374  *
            1375  **        CONVERT INTEGER TO FLOATING INTO FAC AREA
            1376  *
6BDC 066D0A 1377            CALL    CLRFAC            CLEAR FAC AREA FIRST
6BDF 8E5D6B 1378            $IF     @ARG+1 .EQ. 0 GOTO SKIP    IF ZERO ...
6BE2 FC
6BE3 BE4A40 1379            ST      >40,@FAC          IF LESS THAN 100
6BE6 865C   1380            CLR     @ARG              PREPARE FOR WORD OPERATION
6BE8 AE5C64 1381            DIV     100,@ARG          DIVIDE BY 100
6BEB BC4B5C 1382            ST      @ARG,@FAC+1       VALUE IF OVER 100
6BEE BC4C5D 1383            ST      @ARG+1,@FAC+2     Move to the third digit.
6BF1 8E4B4B 1384            $IF     @FAC+1 .EQ. 0 THEN   Value is less than 100   1
6BF4 FA
6BF5 C04C4B 1385            EX      @FAC+1,@FAC+2     Move third byte to second  1
6BF8 4B4C   1386            $ELSE                     If over 100                1
6BFA 904A   1387            INC     @FAC              Increment exponent value   1
            1388            $END    IF
            1389  *
            1390  *
            1391  SKIP
6BFC 0F15   1392            XML     ASSGNV            {ASSIGN TO NUMERIC-VARIABLE}
6BFE D642B3 1393            $IF     @CHAT .NE. COMMA$ GOTO RTBSC
6C01 4CFC
6C03 9104   1394            DINC    @PC         {INC POINTER TO NEXT ERAM ADDRESS}
6C05 056B9E 1395            B       PEEK2
            1396  *
            1397  *****
```

```
1399  ******************************************
1400  *                                        *
1401  *  LINK INSTRUCTION : SE September, 1980 *
1402  *                                        *
1403  ******************************************
1404  *
1405  *  FORMAT:
1406  *
1407  *  CALL LINK("file-name",parameter1,parameter2,.....)
1408  *
1409  *  LINK ROUTINE READS THE FILE NAME SPECIFIED BY THE USER
1410  *  AND SAVE THE ADDRESS OF THE NAME FOR LATER USE.
1411  *  THE FILE WILL BE SEARCHED IN UTILITY CODE LATER ON.
1412  *
1413  *  PARAMETERS ARE PASSED EITHER BY REFERENCE OR BY VALUE.
1414  *  NUMERIC OR STRING VARIABLES AND NUMERIC OR STRING ARRAYS
1415  *  ARE PASSED BY REFERENCE AND ALL OTHERS INCLUDING A USER
1416  *  DEFINED FUNCTION ARE PASSED BY VALUE.
1417  *
1418  *  PARAMETE INFORMATION IS STORED IN ERAM >200a THRU >201A
1419  *  THAT GIVES A PARAMETER TYPE CODE OF EACH PARAMETER.
1420  *    CODE 0 ....   Numeric expression
1421  *    CODE 1 ....   String expression
1422  *    CODE 2 ....   Numeric variable
1423  *    CODE 3 ....   String variable
1424  *    CODE 4 ....   Numeric array
1425  *    CODE 5 ....   String array
1426  *
1427  *  IF A PARAMETER IS PASSED AS A NUMERIC EXPRESSION ITS
1428  *  ACTUAL VALUE GETS PUSHED INTO THE VALUE STACK.
1429  *  IN CASE OF A STRING EXPRESSION, ITS VALUE STACK CONTAINS
1430  *  AN ID(>65), POINTER TO THE VALUE SPACE AND ITS LENGTH.
1431  *  IF A PARAMETER GETS PASSED AS A REFERENCE THE PRODUCT
1432  *  OF XML SYM AND XML SMB IN THE @FAC AREA GETS PUSHED INTO
1433  *  STACK.
1434  *
1435  *  AFTER AN ASSEMBLY LANGUAGE SUBPROGRAM IS EXECUTED LINK
1436  *  ROUTINE WILL POP THE STACK TO GET RID OF PARAMETER
1437  *  INFORMATION.  CONTROL WILL BE TRANSFERRED TO THE BASIC
1438  *  MAIN PROGRAM AFTERWARDS.
1439  *
1440  ****  DATA  AREA  ****
1441  *  CPU RAM FREE SPACE USED IN THIS ROUTINE
1442  *
1443  *  FLAG BITS STORED IN LOCATIONS 00 TO >0F
1444  ****
1445  *
```

```
                  ****************************************
                  *  CALL LINK program                   *
                  ****************************************
                  *
                  BLINK
6C08 B64808                 SB      @FLAG,BSC      CALLED FROM BASIC PROGRAM.
6C0B 066AD2                 CALL    SKIPN          SKIP NAME FIRST
6C0E 066504                 CALL    CHKRAM         Make sure E-RAM is there
6C11 D78F9D                 $IF     @IDCODE .DNE. >A55A GOTO ERRPN   No program
6C14 00A55A
6C17 4E02
6C19 864808                 SB      @FLAG,BSC      Set bit for BASIC flag
6C1C BD106E                 DST     @VSPTR,@OLDS   Save VSPTR for later use.
6C1F 0EB6                   PARSE   RPAR$          Get the routine name.
                  *                                Read up to ")".
6C21 D64C65                 $IF     @FAC2 .NE. STRING GOTO ERRSNM  Should be string
6C24 4E20
6C26 C65106                 $IF     @FAC+6 .H. 6 GOTO ERRBA  Should be less than 6 c
6C29 6E25
6C2B 0F17                   XML     VPUSH         Push to make it semi-permanent.
6C2D 8612                   CLR     @CNT          Initialize parameter counter.
6C2F BF169D                 DST     PARM,@PCODE   Set initial address for parameter
6C32 0A
                  *
                  ****************************************
                  *  PARAMETERS get evaluated here       *
                  ****************************************
                  PAR01
6C33 D642B6                 $IF     @CHAT .EQ. RPAR$ GOTO EXE01  No arg. So execute
6C36 6CD5
6C38 D642B3                 $IF     @CHAT .NE. COMMA$ GOTO ERRIS  Should have a com
6C3B 4E07
                  *
6C3D BD222C                 DST     @PGMPTR,@ERRCOD   Save text pointer.
6C40 0F1B                   XML     PGMCH             Get the character
6C42 CA4280                 $IF     @CHAT .HE. >80 GOTO VAL01  Must be an expressio
6C45 6C98
                  *         $IF     @CHAT .EQ. LPAR$   Pass by expression.
6C47 066D0A                 CALL    CLRFAC            Clear FAC entry for SYM.
6C4A 0F13                   XML     SYM               Read in the symbol table info
                  *
                  *  After XML SYM @FAC area contains a pointer to
                  *  symbol table.
                  *
                  *  Below statement checks if it is a UDF.
6C4C DAB04A                 $IF     .BIT6 RAM(@FAC) .EQ. 1 GOTO VAL01  Pass by valu
6C4F 404C98
6C52 D642B3                 $IF     @CHAT .EQ. COMMA$ GOTO REF01   Pass by reference
6C55 6CB1
6C57 D642B6                 $IF     @CHAT .EQ. RPAR$ GOTO REF01    Pass by reference
6C5A 6CB1
6C5C D642B7                 $IF     @CHAT .EQ. LPAR$ GOTO ARRAY    An array.
6C5F 6C68
6C61 CA4280                 $IF     @CHAT .HE. >80 GOTO VAL01      Pass by value.
6C64 6C98
```

```
6C66 4E07   1489          BR      ERRIS
            1490  *
            1491  ***************************************
            1492  *  ARRAY case gets checked here        *
            1493  ***************************************
            1494  *  should look like A(,,) etc.
            1495  *  Stack entry for an array will look like
            1496  *  -------------------------------------------------------
            1497  *  | Pointer to      |>00  |     | Pointer to  |           |
            1498  *  | symbol table    | or  |     | dim info in |           |
            1499  *  | entry           |>65  |     | real v.s.   |           |
            1500  *  FAC---------------FAC2--------FAC4----------FAC6---------
            1501  *
            1502  *
            1503  ARRAY
6C68 0F1B   1504          XML     PGMCH             Get the next character.
6C6A D642B6 1505          $IF     @CHAT .EQ. RPAR$ GOTO ARRAY2   Pass by reference
6C6D 6C7B
6C6F D642B3 1506          $IF     @CHAT .EQ. COMMA$ GOTO ARRAY   More array inform
6C72 6C68
6C74 932C   1507          DDEC    @PGMPTR        Adjust the pointer.
6C76 BE42B7 1508          ST      LPAR$,@CHAT
6C79 4CB1   1509          BR      REF01          Pass by reference.
            1510  *
            1511  ARRAY2
            1512  *
            1513  *  In array cases the symbol table address gets stored
            1514  *  at FAC area, and the pointer value space
            1515  *  (dimension info.) goes into FAC4
            1516  *
6C7B 0F1B   1517          XML     PGMCH          Advance the program pointer.
6C7D DAB04A 1518          $IF     .BIT7 RAM(@FAC) .NE. 1 THEN   Test string bit 1
6C80 804C8A
6C83 BEC000 1519          ST      4,@0(PCODE)    Numeric array.                1
6C86 1604
6C88 4C8F   1520          $SELSE                                               1
6C8A BEC000 1521          ST      5,@0(PCODE)    String array case.            1
6C8D 1605
            1522          $END    IF
            1523  *
            1524  *  Put the pointer to the value space (dimension info.)
            1525  *  into FAC4.
6C8F BD4E4A 1526          DST     @FAC,@FAC4     Array is not shared.
6C92 A34E00 1527          DADD    6,@FAC4        Point to value space
6C95 06
            1528  *
6C96 4CC8   1529          BR      PUSH
            1530  *
            1531  ***************************************
            1532  *  VALUE                               *
            1533  *  Passing the parameter by value      *
            1534  ***************************************
            1535  VAL01
6C98 BD2C22 1536          DST     @ERRCOD,@PGMPTR   Restore program pointer.
6C9B 0F18   1537          XML     PGMCH             Skip the first character.
```

```
6C9D 0EB6   1538          PARSE   RPAR$            Parsing up to comma.
            1539  *
            1540  *  After parsing @FAC area contains
            1541  *  its actual numeric value in a numeric case, and the
            1542  *  following information in a string case.
            1543  *  ----------------------------------------------------------
            1544  *  | >001C           |>65  |     | Pointer to  | Length of |
            1545  *  | or value pointer|     |     | string      | string    |
            1546  *  | address         |     |     |             |           |
            1547  *  FAC---------------FAC2--------FAC4----------FAC6---------
            1548  *
            1549  *
6C9F D64C65 1550          $IF     @FAC2 .EQ. STRING THEN     If it is a string  1
6CA2 4CAB
6CA4 BEC000 1551          ST      1,@0(PCODE)  Store flag for string expression1
6CA7 1601
6CA9 4CAF   1552          $SELSE                                                1
6CAB 86C000 1553          CLR     @0(PCODE)   Otherwise it is a numeric express 1
6CAE 16
            1554          $END    IF
6CAF 4CC8   1555          BR      PUSH         Push into stack.
            1556  *
            1557  ***************************************
            1558  *  REFERENCE                          *
            1559  *  Passing the parameter by reference  *
            1560  ***************************************
            1561  *  Variables, array element and whole array passing.
            1562  *
            1563  *
            1564  *  After SMB @FAC Entry should look like
            1565  *  ----------------------------------------------------------
            1566  *  | Pointer to      |>00  |     | Pointer to  |           |
            1567  *  | symbol table    |     |     | value space |           |
            1568  *  | entry           |     |     |             |           |
            1569  *  FAC---------------FAC2--------FAC4----------FAC6---------
            1570  *  for a numeric case, and
            1571  *  ----------------------------------------------------------
            1572  *  | Pointer to      |>65  |     | Pointer to  | String    |
            1573  *  | value space     |     |     | string      | length    |
            1574  *  | entry           |     |     |             |           |
            1575  *  FAC---------------FAC2--------FAC4----------FAC6---------
            1576  *  for a string case.
            1577  *
            1578  REF01
6CB1 0F14   1579          XML     SMB              Get the location.
6CB3 CA42B8 1580          $IF     @CHAT .HE. >B8 GOTO VAL01  Pass array expressio
6CB6 6C98
6CB8 8E4C4C 1581          $IF     @FAC2 .EQ. 00 THEN                            1
6CBB C3
6CBC BEC000 1582          ST      2,@0(PCODE)      Must be a numeric variable  1
6CBF 1602
6CC1 4CC8   1583          $SELSE                                                1
6CC3 BEC000 1584          ST      3,@0(PCODE)      Must be a string variable   1
6CC6 1603
            1585          $END    IF
```

```
            1586  *
            1587  ****************************************
            1588  *  PUSH routine                        *
            1589  *  Pushes @FAC entry into a value stack*
            1590  ****************************************
            1591  PUSH
6CC8 9012   1592          INC    @CNT              Increment number of parameter
6CCA C61210 1593          $IF    @CNT .H. 16 GOTO ERRBA   Too many parameters.
6CCD 6E25
6CCF 9116   1594          DINC   @PCODE            Increment parm info address
            1595  *
6CD1 0F17   1596          XML    VPUSH
6CD3 4C33   1597          BR     PAR01             Get the next argument.
            1598  *
            1599  ****************************************
            1600  *  EXECUTE routine                     *
            1601  *  Restore file name info and transfer *
            1602  *  control over to ALC                  *
            1603  ****************************************
            1604  EXE01
6CD5 BE4A20 1605          ST     SPACE,@FAC        Store blank in the FAC area.
6CD8 350005 1606          MOVE   5 FROM @FAC TO @FAC+1
6CDB 4BAA
6CDD 350004 1607          MOVE   4 FROM RAM(12(OLDS)) TO @TEMP
6CE0 00E00C
6CE3 10
LAST ERROR FOUND AT LINE  1305
***** ERROR IN IMMEDIATE FIELD  LINE= 11608  COLUMN= 33
***** ERROR IN BRANCH ADDRESS  LINE= 1608
6CE4 8F0240 1608          $IF    @TEMP+2 .DEQ. 0 GOTO EX10   Do not move 0 bytes
6CE7 00
6CE8 34024A 1609          MOVE   @TEMP+2 FROM RAM(@TEMP) TO @FAC  Mv name to FAC
6CEB B000
6CED BD5002 1610          DST    @TEMP+2,@FAC6   Copy name length
LAST ERROR FOUND AT LINE  1608
***** ERROR IN IMMEDIATE FIELD  LINE= 1611  COLUMN= 13
***** ERROR IN BRANCH ADDRESS  LINE= 1611
6CF0 0000   1611          BR     EX10          Join execution code.
            1612  *
            1613  ****************************************
            1614  *  RETURN to the BASIC main program.   *
            1615  ****************************************
            1616  NOERR
6CF2 C56E10 1617          $WHILE @VSPTR .DH. @OLDS                            1
6CF5 4CFC
6CF7 0F18   1618          XML    VPOP          Pop the stack.                   1
6CF9 056CF2 1619          $END   WHILE         Keep popping till no stack for
            1620  *
            1621  RTBSC
6CFC D642B6 1622          $IF    @CHAT .NE. RPAR$ GOTO ERRIS   Check ")".
6CFF 4E07
            1623  RTBSC2
6D01 0F1B   1624          XML    PGMCH         Advance to the next character.
6D03 BE424E 1625          $IF    @CHAT .NE. 00 GOTO ERRIS  Check end of statemen
6D06 07
6D07 060012 1626          CALL   RPL           Go back to BASIC
```

```
            1628  ****************************************
            1629  *  SUBROUTINES used in this file.      *
            1630  ****************************************
            1631  CLRFAC
6D0A 864A   1632            CLR     @FAC
6D0C 350007 1633            MOVE    7 FROM @FAC TO @FAC+1
6D0F 4B4A
6D11 00     1634            RTN
            1635
```

```
                *
                ****************************************
                *                                      *
                *  CHARPAT ROUTINE                     *
                *                                      *
                ****************************************
                *
                * FORMAT:
                *  CALL CHARPAT open ( <numeric expression> comma
                *                   <string expression> )* close
                * FUNCTION:
                *  RETURNS THE CHARACTER DEFINITION PATTERN FOR CHARACTER
                *  NUMBER <numeric expression> INTO <string expression>.
                *
                ****
     0010       TBLPTR    EQU     >10           Table pointer
     0012       STRPTR    EQU     >12           String pointer
                ****
                GETCHR
6D12 066AD2                CALL    SKIPN         Skip subroutine name
                GCHR2
6D15 0EB6                  PARSE   RPAR$
6D17 D64C65                $IF     @FAC2 .EQ. STRING GOTO ERRSNM  Can't be a strin
6D1A 6E20
6D1C 0F12                  XML     CFI           Convert FAC to integer
                *  Note that 32 - 159 char are supported in Console BASIC
6D1E D65403                $IF     @FAC10 .EQ. 3 GOTO ERRNTB   {range 32 -> 159}
6D21 6E1B
6D23 D34A00                $IF     @FAC .DLT. #32 GOTO ERRBA      "     "     "
6D26 204E25
6D29 CF4A00                $IF     @FAC .DGT. #159 GOTO ERRBA     "     "     "
6D2C 9F6E25
62DF E34A00                DSLL    @FAC,3        {8 bytes/entry so multiply by 8}
6D32 03
6D33 BF1003                DST     >300,@TBLPTR  {Base of character table - 32*8}
6D36 00
6D37 A1104A                DADD    @FAC,@TBLPTR  {Add in arg offset}
                *
6D3A BF0C00                DST     16,@BYTE  Get a 16 byte string in string space
6D3D 10
6D3E 060038                CALL    GETSTR
6D41 BD121C                DST     @SREF,@STRPTR   Save pointer to string
6D44 BE0408                ST      8,@TEMP4        Loop counter
                **
                GETLP
6D47 BC00B0                ST      RAM(@TBLPTR),@TEMP
6D4A 10
6D4B E70000                DSRL    @TEMP,4       Shift 4 bits to right.
6D4E 04
6D4F A20030                ADD     >30,@TEMP     Add ASCII "0"
6D52 CE0039                $IF     @TEMP .LE. >39 GOTO GCHR3 >39=ASCII{"9"}
6D55 4D5A
6D57 A20007                ADD     7,@TEMP       Value "A" -> "F"
                GCHR3
6D5A E60104                SRL     @TEMP+1,4     Shift 4 bits for next byte.
```

```
6D5D A20130 1682          ADD    >30,@TEMP+1   Add ASCII "0"
6D60 CE0139 1683          $IF    @TEMP+1 .LE. >39 GOTO GCHR4
6D63 4D68
6D65 A20107 1684          ADD    7,@TEMP+1     Value "A" -> "F"
            1685  GCHR4
6D68 BDB012 1686          DST    @TEMP,RAM(@STRPTR)   Now put in RAM
6D6B 00
6D6C 9110   1687          DINC   @TBLPTR       Next byte
6D6E 9512   1688          DINCT  @STRPTR       Move pointer two bytes
6D70 9204   1689          DEC    @TEMP4        Count down
6D72 4D47   1690          BR     GETLP         Continue loop
            1691  *
            1692  * NOW ASSIGN THE STRING JUST CREATED TO THE STRING
            1693  * VARIABLE FOLLOWING
            1694  ***                                                  ***
            1695  ***                                                  ***
6D74 0F1B   1696          XML    PGMCH         {SKIP ","}
6D76 CA4280 1697          $IF    @CHAT .HE. >80 GOTO ERRIS  {DON'T ALLOW TOKEN}
6D79 6E07
6D7B 0F13   1698          XML    SYM           {GET SYMBOL NAME}
6D7D 0F14   1699          XML    SMB           {GET VALUE POINTER}
6D7F 0F17   1700          XML    VPUSH         {SAVE FAC ON STACK FOR ASSGNV}
            1701  ***
6D81 D64C65 1702          $IF    @FAC2 .NE. STRING GOTO ERRSNM  Must be string v
6D84 4E20
6D86 BF4A00 1703          DST    >001C,@FAC   Temp string so use SREF as address
6D89 1C
6D8A BD4E1C 1704          DST    @SREF,@FAC4  Pointer to string
6D8D BF5000 1705          DST    16,@FAC6     String length
6D90 10
6D91 0F15   1706          XML    ASSGNV       Assign to string variable
6D93 D642B3 1707          $IF    @CHAT .NE. COMMA$ GOTO RTBSC   Finish looping!
6D96 4CFC
6D98 0F1B   1708          XML    PGMCH        Advance pointer, Skip ","
6D9A 4D15   1709          BR     GCHR2        Repeat the process
            1710  *
```

```
                *****
                ***************************************************************
                * ERROR BRANCH TABLE FOR LINK                                 *
                ***************************************************************
                *
                *  ERROR MESSAGE EQUATES (msg are in console GROM)
                *
202C            MSG1      EQU     >202C         Incorrect statement
2040            MSG4      EQU     >2040         Bad name
2049            MSG5      EQU     >2049         Memory full
2055            MSG6      EQU     >2055         Can't continue
                MSG3
                MSG14
2064            MSG7      EQU     >2064         Bad value
206E            MSG8      EQU     >206E         Number too big
207D            MSG11     EQU     >207D         String-number mismatch
2094            MSG12     EQU     >2094         Bad argument
20A1            MSG13     EQU     >20A1         Bad subscript
20AF            MSG15     EQU     >20AF         Name conflict
                MSG2
20BD            MSG17     EQU     >20BD         Can't do that
20D9            MSGBLN    EQU     >20D9         Bad line number
20F9            MSG18     EQU     >20F9         For-next error
2113            MSG21     EQU     >2113         I/O error
211D            MSG22     EQU     >211D         File error
2128            MSG23     EQU     >2128         Input error
2134            MSG24     EQU     >2134         Data error
213F            MSG19     EQU     >213F         Line too long
                *
                ****   ERROR BRANCHES        *****
                *
                ERRBSC
6D9C 9673                 DECT    @SUBSTK       Adjust subroutine stack
6D9E CA2208                 $IF     @ERRCOD .L. 8 GOTO ERRDSR   I/O error
6DA1 4E61
6DA3 CA220F                 $IF     @ERRCOD .HE. >F GOTO ERRORB   Do not close file
6DA6 6DAB
6DA8 0668C4                 CALL    CLSNO         Close the file
                ERRORB
6DAB C62221                 $IF     @ERRCOD .H. >21 GOTO ERRUEC   Unknown error
6DAE 6E5C
6DB0 A62208                 SUB     8,@ERRCOD
6DB3 8A22                   CASE    @ERRCOD
6DB5 4DE9                   BR      ERRMM          8   Memory full
6DB7 4E07                   BR      ERRIS          9   Control char - not issued
6DB9 4DEE                   BR      ERRTAG         A   Bad tag
6DBB 4DF3                   BR      ERRCHK         B   Checksum error
6DBD 4DF8                   BR      ERRDD          C   Duplicate definition
6DBF 4DFD                   BR      ERRUR          D   Unresolved reference
6DC1 4E07                   BR      ERRIS          E   No entry address -- not issue
6DC3 4E02                   BR      ERRPN          F   Program not found
6DC5 4E07                   BR      ERRIS         10   Incorrect statement
6DC7 4E0C                   BR      ERRBN         11   Bad name
6DC9 4E11                   BR      ERRCC         12   Can't continue
6DCB 4E16                   BR      ERRBV         13   Bad value
```

```
6DCD 4E1B   1764            BR      ERRNTS        14   Number too big
6DCF 4E20   1765            BR      ERRSNM        15   String-number mismatch
6DD1 4E25   1766            BR      ERRBA         16   Bad argument
6DD3 4E2A   1767            BR      ERRBS         17   Bad subscript
6DD5 4E2F   1768            BR      ERRNC         18   Name conflict
6DD7 4E34   1769            BR      ERRCDT        19   Can't do that
6DD9 4E39   1770            BR      ERRBLN        1A   Bad line number
6DDB 4E3E   1771            BR      ERRFNE        1B   For-next error
6DDD 4E68   1772            BR      ERRIOB        1C   I/O error
6DDF 4E43   1773            BR      ERRFE         1D   File error
6DE1 4E48   1774            BR      ERRINP        1E   Input error
6DE3 4E4D   1775            BR      ERRDAT        1F   Data error
6DE5 4E52   1776            BR      ERRLTL        20   Line too long
6DE7 4DE9   1777            BR      ERRMM         21   Memory full without
            1778  *                                      file being open
            1779  ****************************************
            1780  *  ERROR HANDLING SECTION              *
            1781  ****************************************
            1782  ****
6DE9 06001C 1783  ERRMM     CALL    ERR$$         * Memory full
6DEC 2049   1784            DATA    #MSG5
            1785  *
6DEE 06001C 1786  ERRTAG    CALL    ERR$$         * Bad tag
6DF1 6E6D   1787            DATA    #MSGTAG
            1788  *
6DF3 06001C 1789  ERRCHK    CALL    ERR$$         * Checksum error
6DF6 6E79   1790            DATA    #MSGCHK
            1791  *
6DF8 06001C 1792  ERRDD     CALL    ERR$$         * Duplicate definition
6DFB 6E88   1793            DATA    #MSGDD
            1794  *
6DFD 06001C 1795  ERRUR     CALL    ERR$$         * Unresolved references
6E00 6E9D   1796            DATA    #MSGUR
            1797  *
6E02 06001C 1798  ERRPN     CALL    ERR$$         * PROGRAM NOT FOUND
6E05 6EB2   1799            DATA    #MSGPN
            1800  *
6E07 06001C 1801  ERRIS     CALL    ERR$$         * Incorrect statement
6E0A 202C   1802            DATA    #MSG1
            1803  *
6E0C 06001C 1804  ERRBN     CALL    ERR$$         * BAD NAME
6E0F 2040   1805            DATA    #MSG4
            1806  *
6E11 06001C 1807  ERRCC     CALL    ERR$$         * Can't continue
6E14 2055   1808            DATA    #MSG6
            1809  *
6E16 06001C 1810  ERRBV     CALL    ERR$$         * BAD VALUE
6E19 2064   1811            DATA    #MSG7
            1812  *
6E1B 06001C 1813  ERRNTB    CALL    ERR$$         * NUMBER TOO BIG
6E1E 206E   1814            DATA    #MSG8
            1815  *
6E20 06001C 1816  ERRSNM    CALL    ERR$$         * String Number mismatch
6E23 2070   1817            DATA    #MSG11
            1818  *
```

```
6E25 06001C 1819  ERRBA     CALL    ERR$$         * BAD ARGUMENT
6E28 2094   1820            DATA    #MSG12
            1821  *
6E2A 06001C 1822  ERRBS     CALL    ERR$$         * BAD SUBSCRIPT
6E2D 20A1   1823            DATA    #MSG13
            1824  *
6E2F 06001C 1825  ERRNC     CALL    ERR$$         * NAME CONFLICT
6E32 20AF   1826            DATA    #MSG15
            1827  *
6E34 06001C 1828  ERRCDT    CALL    ERR$$         * CAN'T DO THAT
6E37 20BD   1829            DATA    #MSG17
            1830  *
6E39 06001C 1831  ERRBLN    CALL    ERR$$         * Bad line number
6E3C 20D9   1832            DATA    #MSGBLN
            1833  *
6E3E 06001C 1834  ERRFNE    CALL    ERR$$         * For-next error
6E41 20F9   1835            DATA    #MSG18
            1836  *
6E43 06001C 1837  ERRFE     CALL    ERR$$         * File error
6E46 211D   1838            DATA    #MSG22
            1839  *
6E48 06001C 1840  ERRINP    CALL    ERR$$         * Input error
6E4B 2128   1841            DATA    #MSG23
            1842  *
6E4D 06001C 1843  ERRDAT    CALL    ERR$$         * Data error
6E50 2134   1844            DATA    #MSG24
            1845  *
6E52 06001C 1846  ERRLTL    CALL    ERR$$         * Line too long
6E55 213F   1847            DATA    #MSG19
            1848  *
6E57 06001C 1849  ERRRMB    CALL    ERR$$         * No memory expansion
6E5A 6EC4   1850            DATA    #MSGRAM
            1851  *
6E5C 06001C 1852  ERRUEC    CALL    ERR$$         * Unknown error code
6E5F 6EDB   1853            DATA    #MSGUEC
            1854  *
            1855  ********
            1856  ****************************************************************
            1857  * ERRDSR -- routine to handle IO errors for BASIC             *
            1858  ****************************************************************
            1859  ERRDSR
6E61 BD041C 1860            DST     @SREF,@TEMP4  Copy PAB address
6E64 A70400 1861            DSUB    04,@TEMP4     BASIC assumes offset by 4
6E67 04
            1862  *
6E68 06001C 1863  ERRIOB    CALL    ERR$$
6E6B 2113   1864            DATA    #MSG21        * I/O error
            1865  **
            1866  ********
            1867  ********
            1868  ****  Error messages
            1869  *******
            1870            BASE    0,0,0,0,0,0,>60
6E6D 0BA9AC 1872  MSGTAG    DATA    11,:ILLEGAL TAG:
6E79 0EA3A8 1873  MSGCHK    DATA    14,:CHECKSUM ERROR:
```

```
6E88 14A4B5 1874  MSGDD    DATA   20,:DUPLICATE DEFINITION:
6E9D 14B5AE 1875  MSGUR    DATA   20,:UNRESOLVED REFERENCE:
6EB2 11B0B2 1876  MSGPN    DATA   17,:PROGRAM NOT FOUND:
6EC4 13AEAF 1877  MSGRAM   DATA   19,:NO MEMORY EXPANSION:
6ED8 12B5AE 1878  MSGUEC   DATA   18,:UNKNOWN ERROR CODE:
     7000   1880  ALCODE   EQU    >7000
            1881  *   END
```

```
SOURCE ACCESS NAME=     PCD2.ASSM.SRC.BSCSUP2
OBJECT ACCESS NAME=     PCD2.ASSM.OBJ.BSCSUP2
LISTING ACCESS NAME=    PCD2.ASSM.LST.BSCSUP2
ERROR ACCESS NAME=      PCD2.ATA.LST.ERRS
OPTIONS=                XREF,BUNLST,TUNLST,DUNLST
MACRO LIBRARY PATHNAME=
```

```
                    IDT    'BSCSUP'
            *
            *       BASIC SUPPORT ROUTINES: contain numeric reference and
            *       assignment (NUMREF,NUMASG), string reference and assignmen
            *       (STRREF,STRASG), and error reporting (ERR).
            *
      00CE  SET     EQU    >00CE          Works with console, not debugger cons
            *
            *       SYSTEM EQUATES
            *
      000E  SCNKEY  EQU    >000E
      6010  EXBASX  EQU    >6010          EXTENDED BASIC XML TABLES (BASE)
      200A  PARM    EQU    >200A          Information for parameter type.
      8300  PAD     EQU    >8300
      830C  BYTE    EQU    >830C
      8310  OLDS    EQU    >8310          OLD VALUE PTR BEFORE EVALUATE SUB
            *                             ARGUMENT
      8312  COUNT   EQU    >8312          NO. OF ARGUMENT
      831C  SREF    EQU    >831C
      8322  ERRCOD  EQU    >8322          PLACE TO RETURN ERROR CODE
      8343  BASE    EQU    >8343
      8348  FLAG    EQU    >8348          Flag used in GPL main program.
      834A  FAC     EQU    >834A
      8354  SCTEMP  EQU    >8354
      8355  SCLEN   EQU    >8355
      8356  SCNAME  EQU    >8356
      8350  FAC6    EQU    FAC+6
      836E  VSPTR   EQU    >836E          VALUE STACK PTR
      8370  MAXVDP  EQU    >8370
      837C  STATUS  EQU    >837C
      83CC  SNDADD  EQU    >83CC
      83CE  STFLGS  EQU    >83CE
      83D0  CRULST  EQU    >83D0
      83E0  GPLWS   EQU    >83E0          GPL/EXTENDED BASIC WORKSPACE
            *
      4000  WRVDP   EQU    >4000          Write enable for VDP
      8800  VDPRD   EQU    >8800          VDP read data address
      8C00  VDPWD   EQU    >8C00          VDP write data address
      8C02  VDPWA   EQU    >8C02          VDP write address address
            *
            *       ERROR EQUATES
            *
      1000  ERRIS   EQU    >1000          Incorrect statement
      1100  ERRBN   EQU    >1100          Bad name
      2100  ERRMN   EQU    >2100          Memory full
      1200  ERRCC   EQU    >1200          Can't continue
      1300  ERRBV   EQU    >1300          Bad value
      1400  ERRNTB  EQU    >1400          Number too big
      1500  ERRSNM  EQU    >1500          String number mismatch
      1600  ERRBA   EQU    >1600          Bad argument
      1700  ERRBS   EQU    >1700          Bad subscript
      1800  ERRNC   EQU    >1800          Name conflict
      1900  ERRCDT  EQU    >1900          Can't do that
      1A00  ERRBLN  EQU    >1A00          Bad line number
      1B00  ERRFNE  EQU    >1B00          For next error
      1C00  ERRIOB  EQU    >1C00          I/O error
      1D00  ERRFE   EQU    >1D00          File error
      1E00  ERRINP  EQU    >1E00          Input error
      1F00  ERRDAT  EQU    >1F00          Data error
      2000  ERRLTL  EQU    >2000          Line too long
```

```
0061      2200  ERRSYN  EQU    >2200          Syntax error
0062      2300  ERRNO   EQU    >2300          Numeric overflow
0063      2400  ERRUC   EQU    >2400          Unrecognized character
0064      0F00  ERRSNF  EQU    >0F00          Program not found
0065      2500  ERRST   EQU    >2500          String truncated
0066            *
0067            *
0068            *       UTILITY BLWP VECTORS
0069            *
0070                    REF    GPLLNK,UTLTAB
0071                    DEF    NUMASG,NUMREF,STRASG,STRREF,ERR
0072 0000 0014' NUMASG  DATA   UTILWS,NAENTR     Numeric assignment
0073 0004 0014' NUMREF  DATA   UTILWS,NRENTR     Numeric reference
0074 0008 0014' STRASG  DATA   UTILWS,SAENTR     String assignment
0075 000C 0014' STRREF  DATA   UTILWS,SRENTR     String reference
0076 0010 0014' ERR     DATA   UTILWS,ERENTR     Return error code to BASIC
0077            *
0078            *       UTILITY WORKSPACE
0079            *
0080 0014 0000  UTILWS  DATA   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0081      0019' R2LB    EQU    UTILWS+5
0082      001D' R4LB    EQU    UTILWS+9
0083            *
0084            ** D A T A    C O N S T A N T S
0085            *
0086 0034   65  CBH65   BYTE   >65
0087 0035   20  EQBIT   BYTE   >20
0088            *
```

```
                 *
                 *      NUMERIC ASSIGNMENT UTILITY
                 *
                 *
                 *        UTILITY ROUTINE NAENTR
                 *         A utility to allow a numeric value to be assigned
                 *         to a numeric variable or a numeric array element
                 *         passed in a parameter list of LINK call.
                 *         The numeric value to be assigned should be in FAC a
                 *         The stack has info. For each argument.
                 *         For a simple numeric variable, the stack entry looks
                 *         like this :
                 *              ----------------------------------------
                 *              |Ptr to     |>00|  |Ptr to      |       |
                 *              |S.T. entry |   |  |Value space |       |
                 *              ----------------------------------------
                 *          For an array element, the stack entry looks like:
                 *             ----------------------------------------
                 *             | Ptr to     |  |  |Ptr to     |        |
                 *             | S.T. entry |  |  |Dim. Info. |        |
                 *             ----------------------------------------
                 *         REGISTER USAGE
                 *          R0 : Array element no. or zero
                 *          R1 : Argument number
                 *          R0,R1,R2,R3,R4,R5,R6,R7,R8 are used in this
                 *          routine
                 *
                 *          5 types of arguments :
                 *  Identifier 0 : Numeric Expression
                 *  Identifier 1 : String Expression
                 *  Identifier 2 : Numeric Variable
                 *  Identifier 3 : String Variable
                 *  Identifier 4 : Numeric Array
                 *  Identifier 5 : String Array
                 *
                 *
0036 C01D        NAENTR  MOV    *R13,R0          For the change of the work space
0038 C06D                MOV    @2(R13),R1       For the change of the work space
     003A 0002
003C 06A0                BL     @CKARG           Check valid argument no. and
     003E 007E'
                 ************** set up stack entry ptr in R5  **************
                 *******  Load parameter information address >200A *********
                 ****Check  SIMPLE NUMERIC VAR. OR AN ARRAY REF.   *********
0040 D0D3                MOVB   *R3,R3           Get the one byte identifier in
                 *                               CPU
0042 1368                JEQ    NUERR2           Can't do the assignment to a
                 *                               numeric expression argument
0044 0983                SRL    R3,8             Make it double
0046 0643                DECT   R3               Flag 2 is for NUMERIC VARIABLE
0048 1617                JNE    NUM04            No - go to the code for array
                 ******  FOR ASSIGNMENT TO A SIMPLE NUMERIC VAR.     *********
004A C000                MOV    R0,R0            In this case, R0Mmust contain 0
004C 1663                JNE    NUERR2           ERROR : BAD ARGUMENT
004E C0C5                MOV    R5,R3            Get the ptr to the stack entry
0050 05C3                INCT   R3               Point to the ID byte of the
                 *                               stack entry
0052 06A0                BL     @GETV1           Get 1 byte (ID) from VDP
     0054 036C'
                 *                               Numeric ID byte must be 0
```

```
0147 0056 165A          JNE    NUERR1            ERROR:STRING-NO. MISMATCH
0148 0058 05C3          INCT   R3                Point to the value space ptr
0149 005A 06A0          BL     @GET1             Get value address
     005C 0330'
0150 005E C0C1          MOV    R1,R3             Just copy
0151 0060 C103  NUM00   MOV    R3,R4             PUT routine uses R4
0152 0062 0203          LI     R3,FAC            Load FAC address
     0064 834A
0153 0066 0202  NUM01   LI     R2,4              8 bytes to copy
     0068 0004
0154 006A C073  NUM02   MOV    *R3+,R1           Get data from FAC area
0155 006C 06A0          BL     @PUT1             Move two bytes at a time.
     006E 034C'
0156 0070 05C4          INCT   R4                Increment VDP address
0157 0072 0602          DEC    R2                Decrement the counter - done?
0158 0074 15FA          JGT    NUM02             No - loop for more
0159 0076 0380          RTWP                     Yes - return to the caller
0160            *****   FOR ASSIGNMENT TO ONE ELEMENT OF AN ARRAY ***********
0161            *****   R0 : ARRAY ELEMENT NO. R5: PTR TO STACK ENTRY *******
0162 0078 06A0  NUM04   BL     @ARYBND           Check array element w/in bound
     007A 00A2'
0163            *                            and find the ptr to element in ERAM
0164 007C 10F1          JMP    NUM00             Go to the general code to move
0165            *                                from FAC to ERAM
0166            ************************************************************
0167            *******   CHECK VALID ARGUMENT NO. AND SET UP STACK ENTRY **
0168            *******   PTR IN R5                                    *****
0169 007E C041  CKARG   MOV    R1,R1          Argument no. can't be 0
0170 0080 1349          JEQ    NUERR2         ERROR : BAD ARGUMENT
0171 0082 0A81          SLA    R1,8           Make one byte
0172 0084 9060          CB     @COUNT,R1      Valid argument?
     0086 8312
0173 0088 1145          JLT    NUERR2         ERROR : BAD ARGUMENT
0174 008A 0981          SRL    R1,8           Make it two byte again
0175 008C C141          MOV    R1,R5          Get the argument no.
0176 008E 0A35          SLA    R5,3           8 bytes per stack entry
0177 0090 0225          AI     R5,8           Skip the 8 bytes file name
     0092 0008
0178            *                             on top of stack
0179 0094 A160          A      @OLDS,R5       Add to the old stack ptr
     0096 8310
0180            *                             R5 : Ptr to the stack entry
0181 0098 C0C1          MOV    R1,R3          Copy argument number
0182 009A 0603          DEC    R3             Offset by 1
0183 009C 0223          AI     R3,PARM        Get address of parm info.
     009E 200A
0184 00A0 045B          RT
0185            ************************************************************
0186            ********* CHECK THE ARRAY ELEMENT NO. IS WITHIN BOUND  *****
0187 00A2 C24B  ARYBND  MOV    R11,R9         Save rtn addr.
0188 00A4 0643          DECT   R3             Flag 4 is for ARRAY
0189 00A6 1632          JNE    NUERR1         NO-Assume string arg.
0190            *                             Error:STRING-NO MISMATCH
0191 00A8 C0C5          MOV    R5,R3          Get the stack entry ptr
0192 00AA 06A0          BL     @GET1          Get the S.T. entry ptr
     00AC 0330'
0193 00AE C0C1          MOV    R1,R3          R1 : Data reg. for GET1
0194            *                             R3 : Addr. Reg. for GET1
0195 00B0 06A0          BL     @GETV1         Get the 1st byte of S.T. entry
     00B2 036C'
```

```
0196 00B4 112B          JLT    NUERR1         String bit is set - ERROR
0197            *                             STRING-NO MISMATCH
0198 00B6 06A0          BL     @ARY1          Go through this general routine
     00B8 00C2'
0199            *** R3 has address of the value space ****
0200 00BA 6004          S      R4,R0          Check the BASE : 0 : O.K.
0201            *                                      BASE : 1 : offset 1
0202 00BC 0A30          SLA    R0,3           8 bytes per element
0203 00BE A0C0          A      R0,R3          Add to the address of value space
0204 00C0 0459          B      *R9
0205            ************************************************************
0206 00C2 C28B  ARY1    MOV    R11,R10        Save RTN addr.
0207 00C4 0A51          SLA    R1,5           R1 still contains 1st byte from
0208            *
0209            *                             func bits
0210 00C6 09D1          SRL    R1,13          Shift back to use as a double
0211            ******  R1:NO. OF DIMENSION, R5: STACK ENTRY PTR   ********
0212 00C8 C201          MOV    R1,R8          Save no. of dimension for later
0213 00CA D120          MOVB   @BASE,R4       Check the BASE
     00CC 8343
0214 00CE 0984          SRL    R4,8           Make it double
0215 00D0 1303          JEQ    NUM08          If BASE=0,INDEX=0 -- it's o.k.
0216 00D2 0600          DEC    R0             Decrement the INDEX
0217 00D4 1123          JLT    NUERR3         BASE=1, INDEX=0---ERROR
0218            *                             BAD SUBSCRIPT
0219 00D6 0580          INC    R0             Restore INDEX
0220 00D8 0206  NUM08   LI     R6,1           Initial value for accumulator
     00DA 0001
0221 00DC C0C5          MOV    R5,R3          Put the stack entry ptr in R3
0222 00DE 0223          AI     R3,4           Try to get the dim. Info. ptr
     00E0 0004
0223 00E2 06A0          BL     @GET1          Go to get the dim. Info. Ptr
     00E4 0330'
0224 00E6 C0C1          MOV    R1,R3          Put it in R3
0225 00E8 0643          DECT   R3             For the following INCT instruct.
0226 00EA 05C3  NUM10   INCT   R3             Point to dimension infor. entry
0227            *                             2-byte maxima per dimension
0228 00EC 06A0          BL     @GET1          Get the dimension maxima in R1
     00EE 0330'
0229 00F0 0581          INC    R1             BASE=0, add 1 offset
0230 00F2 6044          S      R4,R1          BASE=1, O.K.
0231 00F4 3981          MPY    R1,R6          Get the max. array index
0232            *                             R6 : accumulator
0233 00F6 C186          MOV    R6,R6          First 2 bytes must be 0 here
0234 00F8 1611          JNE    NUERR3         else BAD SUBSCRIPT
0235 00FA C187          MOV    R7,R6          Get last two bytes result
0236 00FC 0608          DEC    R8             Decrement the counter
0237            *                             R8 : no. of dimension
0238 00FE 15F5          JGT    NUM10          Loop for more
0239 0100 0606          DEC    R6             BASE=0:element no. must range
0240            *                             from 0 to (R6-1)
0241 0102 A184          A      R4,R6          BASE=1:element no. range 1 -- R6
0242 0104 8180          C      R0,R6          User's INDEX must <= max. INDEX
0243 0106 150A          JGT    NUERR3         NO - ERROR:BAD SUBSCRIPT
0244 0108 05C3          INCT   R3             Contains value space address in
0245            *                             VDP RAM.
0246 010A 045A          B      *R10           RTN
0247            ************************************************************
0248 010C 0200  NUERR1  LI     R0,ERRSNM      STRING-NO. MISMATCH
     010E 1500
```

```
0249 0110 0460          B      @ERROR         Error out
     0112 039E'
0250 0114 0200  NUERR2  LI     R0,ERRBA       BAD ARGUMENT
     0116 1600
0251 0118 0460          B      @ERROR         Error out
     011A 039E'
0252 011C 0200  NUERR3  LI     R0,ERRBS       BAD SUBSCRIPT
     011E 1700
0253 0120 0460          B      @ERROR
     0122 039E'
```

```
                *
                *       NUMERIC REFERENCE UTILITY
                *
                *
                *         UTILITY ROUTINE NRENTR
                *          A utility to allow reference to a numeric exp.
                *          , numeric variable or a numeric array element
                *          passed in a parameter list of a LINK call.
                *          The numeric value will be placed in the FAC
                *          area after calling NRENTR.
                *          The stack has info. For each argument.
                *          For a simple numeric variable, the stack entry looks
                *          like this :
                *              ----------------------------------------
                *              |Ptr to     |>00|  |Ptr to      |       |
                *              |S.T. entry |   |  |Value space |       |
                *              ----------------------------------------
                *          For an array reference, the stack entry looks
                *          like this:
                *              ----------------------------------------
                *              | Ptr to     |  |  |Ptr to     |        |
                *              | S.T. entry |  |  |Dim. Info. |        |
                *              ----------------------------------------
                *          For a numeric exp., the stack entry has the
                *          8 bytes floating points value passed by a parse
                *          routine.
                *         REGISTER USAGE
                *          R0 : Array element no. or 0
                *          R1 : Arrgument number
                *          R0,R1,R2,R3,R4,R5,R6,R7,R8 are used in this
                *          routine
                *
0124 C01D       NRENTR  MOV    *R13,R0        For the change of the work space
0126 C06D               MOV    @2(R13),R1     For the change of the work space
     0128 0002
012A 06A0               BL     @CKARG         Check valid argument no.
     012C 007E'
                *                             and set up stack ptr and parm add
                *******   R5 : Ptr. To the stack entry, R1 : Argument no.
                *******  CHECK THE ARGUMENT IDENTIFIER IN CPU RAM **********
                ***** A NUMERIC EXP., SIMPLE NUMERIC VAR. OR AN ARRAY ******
012E D0D3               MOVB   *R3,R3         Get the one byte identifier in
                *                             CPU
0130 0983               SRL    R3,8           Make it double
0132 160E               JNE    REF00          Not a numeric expression
                ******  A NUMERIC EXP., MOVE THE 8 BYTES STACK ENTRY *******
                ******  FROM VDP TO FAC.                            *******
0134 C000               MOV    R0,R0          R0 in this case must contain 0
0136 1629               JNE    RFERR2         ERROR : BAD ARGUMENT
0138 0202               LI     R2,8           8 bytes to move
     013A 0008
013C 0204               LI     R4,FAC         Destination addr.
     013E 834A
0140 C0C5               MOV    R5,R3          Source addr.: stack entry ptr
0142 06A0       REF03   BL     @GET1          Get 2 bytes from VDP
     0144 0330'
0146 CD01               MOV    R1,*R4+        Put 2 bytes in FAC
0148 05C3               INCT   R3             Update source addr.
014A 0642               DECT   R2             Update the counter
014C 15FA               JGT    REF03          More to move
```

```
0310 014E 0380          RTWP                    Return to caller
0311            ********   A SIMPLE NUMERIC VAR. OR AN ARRAY ****************
0312 0150 0643  REF00   DECT   R3               Flag 2 is for NUMERIC VARIABLE
0313 0152 1616          JNE    REF04            NO - go to the code for array
0314            ******  FOR REFERENCE TO A SIMPLE NUMERIC VAR.      *********
0315 0154 C000          MOV    R0,R0            In this case, R0 must contain 0
0316 0156 1619          JNE    RFERR2           ERROR : BAD ARGUMENT
0317 0158 C0C5          MOV    R5,R3            Get the ptr to the stack entry
0318 015A 05C3          INCT   R3               Point to the ID byte of the
0319            *                               stack entry
0320 015C 06A0          BL     @GETV1           Get 1 byte (ID) from VDP
     015E 036C'
0321            *                               Numeric ID byte must be 0
0322 0160 1612          JNE    RFERR1           ERROR:STRING-NO. MISMATCH
0323 0162 05C3          INCT   R3               Points to the value stack
0324 0164 06A0          BL     @GET1            Get value space address
     0166 0330'
0325 0168 C0C1          MOV    R1,R3            Copy for later use
0326            *
0327 016A 0204  REF01   LI     R4,FAC           Destination addr. Is in FAC
     016C 834A
0328 016E 0202          LI     R2,4             Move 8 bytes value
     0170 0004
0329 0172 06A0  REF02   BL     @GET1            Move 2 bytes from VDP to R1
     0174 0330'
0330 0176 CD01          MOV    R1,*R4+          Move 2 bytes into FAC area
0331 0178 05C3          INCT   R3               Increment VDP address pointer
0332 017A 0602          DEC    R2
0333 017C 15FA          JGT    REF02            Read more
0334 017E 0380          RTWP
0335            *
0336            *****   FOR ASSIGNMENT TO ONE ELEMENT OF AN ARRAY ***********
0337            *****   R0 : ARRAY ELEMENT NO. R5: PTR TO STACK ENTRY *******
0338 0180 06A0  REF04   BL     @ARYBND          Check array elem. Is w/in bound
     0182 00A2'
0339            *              and find the ptr to the array element in ERAM *
0340 0184 10F2          JMP    REF01            Go to the general code to move
0341            *                               from ERAM to FAC
0342            ***************************************************************
0343 0186 0460  RFERR1  B      @NUERR1          STRING-NO. MISMATCH
     0188 010C'
0344 018A 0460  RFERR2  B      @NUERR2          BAD ARGUMENT
     018C 0114'
```

```
                 ************************************************************
                 *
                 *       STRING ASSIGNMENT UTILITY
                 *
                 *
                 *         UTILITY ROUTINE SAENTR
                 *          A utility to allow a string to be assigned to a
                 *          string variable or a string array element passed
                 *          in a parameter list of a LINK call.
                 *          The stack has info. For each argument.
                 *          For a string variable, the stack entry looks like
                 *          this ----------------------------------------
                 *               |Ptr to     |>65|  |Pointer  | String  |
                 *               |Value space|   |  |to string| Length  |
                 *               ----------------------------------------
                 *          For a string array, the stack entry looks like
                 *          this ----------------------------------------
                 *               |Ptr to     |   |  |Ptr to   |         |
                 *               |S. T. entry|   |  |Dim info.|         |
                 *               ----------------------------------------
                 *          USER'S REGISTER USAGE-- following registers must
                 *           be set up prior to the SAENTR call.
                 *           R0 : Array element no. or zero
                 *           R1 : Argument number
                 *           R2 : String address in ERAM. (the first byte
                 *                of the string is the length of the string.)
                 ************************************************************
                 *       THIS ROUTINE REGISTER USAGE (UTILWS)
                 *           R0 : Array element
                 *           R1 : Argument number.
                 *           R2 : Return address for VGASSN
                 *           R5 : Value stack address.
                 *           R9 : String address copied from user's reg 3.
                 *           R10: Return address for ARY1.
                 ************************************************************
0181 018E C01D   SAENTR  MOV    *R13,R0        Array element from user
0182 0190 C06D           MOV    @2(R13),R1     Argument number
     0192 0002
0183 0194 C26D           MOV    @4(R13),R9     String addr. In ERAM
     0196 0004
0184             *
0185 0198 06A0           BL     @CKARG         Check valid argument no. and
     019A 007E'
0186             *                             set up ptr to stack entry in R5
0187             *       R3 has address of parm type info
0188 019C D0D3           MOVB   *R3,R3         Get the one byte identifier in
0189             *                             CPU
0190 019E 0983           SRL    R3,8           Make it double
0191 01A0 0603           DEC    R3             Check if string expression
0192 01A2 1324           JEQ    STERR2         Can't do the assignment to a
0193             *                             string expression argument
0194 01A4 0643           DECT   R3             Flag 3 is for string variable
0195 01A6 163A           JNE    STARY          NO : IS STRING ARRAY ?
0196             ******  ASSIGNMENT TO STRING VARIABLE    *******
0197 01A8 C000           MOV    R0,R0          Array element has to be zero.
0198 01AA 1620           JNE    STERR2         ERROR : BAD ARGUMENT
0199 01AC C0C5           MOV    R5,R3          Addr.reg. for GETV1
0400 01AE 05C3           INCT   R3             Point to the ID byte of the
0401             *                             stack entry
0402 01B0 06A0           BL     @GETV1         Get 1 byte (ID) from VDP
```

```
     01B2 036C'
 0403 01B4 9801         CB     R1,@CBH65      Is this a string variable ?
     01B6 0034'
 0404 01B8 1615         JNE    STERR1         NO - STRING NO. MISMATCH ERROR
 0405            ************************************************************
 0406            ******  PUSH THIS STACK ENTRY ON TOP OF THE STACK FOR ******
 0407            ******  ROUTINE "ASSGNV" TO USE                       ******
 0408            ************************************************************
 0409            ****    MOVE THIS STRING VARIABLE ENTRY TO CPU FAC AREA  ***
 0410            ****    IN ORDER TO DO A VPUSH                           ***
 0411 01BA 0206         LI     R6,8           # of bytes to move
     01BC 0008
 0412 01BE 0204         LI     R4,FAC         Destination addr. In CPU
     01C0 834A
 0413 01C2 C0C5         MOV    R5,R3          Restore stack entry ptr
 0414            *                             R3 : addr. Reg. for GET1
 0415 01C4 06A0  STRO2  BL     @GET1          Get 2 bytes from VDP
     01C6 0330'
 0416 01C8 CD01         MOV    R1,*R4+        Put to FAC area
 0417 01CA 05C3         INCT   R3             Update source addr.
 0418 01CC 0646         DECT   R6             Update the counter
 0419 01CE 15FA         JGT    STRO2          More to move
 0420            ******  DO THE VPUSH, GETSTR, COPY STRING TO STRING SPACE
 0421            ******  AND ASSIGNMENT                    *****************
 0422 01D0 06A0         BL     @VGASSN
     01D2 0246'
 0423            **********    NOW UPDATES THE STACK ENTRY OF THIS    *******
 0424            **********    STRING VARIABLE                        *******
 0425            **********    R5 STILL POINTS TO THE OLD STACK ENTRY *******
 0426            **********    FOR THIS STRING VARIABLE               *******
 0427            **********    (length only, pointer is garbage anyway) *****
 0428 01D4 C105         MOV    R5,R4
 0429 01D6 0224         AI     R4,6           "STRING LENGTH" entry
     01D8 0006
 0430 01DA C060         MOV    @BYTE,R1       Put the length into data reg.
     01DC 830C
 0431            ****** SRL    R1,8           Make it two bytes
 0432 01DE 06A0         BL     @PUT1          Put 2 bytes of data to VDP
     01E0 034C'
 0433 01E2       STRTN
 0434 01E2 0380         RTWP                  Return
 0435            ************************************************************
 0436 01E4 0200  STERR1 LI     R0,ERRSNM      STRING-NO. MISMATCH
     01E6 1500
 0437 01E8 0460         B      @ERROR
     01EA 039E'
 0438 01EC 0200  STERR2 LI     R0,ERRBA       BAD ARGUMENT
     01EE 1600
 0439 01F0 0460         B      @ERROR         Error return
     01F2 039E'
 0440            ************************************************************
 0441            ************ Common routine for STRASG & STRREF to check
 0442            ************ valid array element
 0443 01F4 C088  STARYC MOV    R11,R2         Save RTN addr.
 0444 01F6 0643         DECT   R3             Flag 5 is for string array
 0445 01F8 16F5         JNE    STERR1         ERROR : STRING NO. MISMATCH
 0446 01FA C0C5         MOV    R5,R3          Get the stack entry ptr
 0447 01FC 06A0         BL     @GET1          Get the S.T. entry ptr
     01FE 0330'
 0448 0200 C0C1         MOV    R1,R3          R1 : Data reg. for GET1
```

```
0449            *                               R3 : Addr. Reg. for GET1
0450 0202 06A0          BL     @GETV1           Get the 1st byte of S.T. entry
     0204 036C'
0451 0206 1102          JLT    STARY1           O.K.
0452 0208 0460          B      @STERR1          ERROR : STRING-NO. MISMATCH
     020A 01E4'
0453 020C       STARY1
0454            *   R0 has array element number
0455 020C 06A0          BL     BARY1            Go through general routine to
     020E 00C2'
0456            *              check array element is out of bound
0457            *  R4:BASE, R0:ARRAY ELEMENT, R3:PTR TO BEGINNING OF VALUE
0458            *                                                  SPACE
0459 0210 6004          S      R4,R0            Check the BASE : 0 : O.K.
0460            *                                         BASE : 1 : Offset 1
0461 0212 0A10          SLA    R0,1             2 bytes per element
0462 0214 A0C0          A      R0,R3            Ptr to the exact element we want
0463 0216 06A0          BL     @GET1            Get the string ptr
     0218 0330'
0464 021A 0452          B      *R2              RTN
0465            ************************************************************
0466            ************************************************************
0467 021C 06A0  STARY   BL     @STARYC          Go through general routine
     021E 01F4'
0468            *                               shared by STRREF
0469            *   NOW  R1 : PTR TO THE OLD STRING
0470            *        R3 : PTR TO THE VALUE SPACE
0471            *** SET UP FAC ENTRY AND PUSH IT ON STACK FOR ASSIGN ROUT.  *
0472 0220 0206          LI     R6,FAC           Destination addr.
     0222 834A
0473 0224 CD83          MOV    R3,*R6+          Put ptr to value space in
0474 0226 DDA0          MOVB   @CBH65,*R6+      Put >65 ID in
     0228 0034'
0475 022A DD84          MOVB   R4,*R6+          Clear the unused byte
0476 022C CD81          MOV    R1,*R6+Put the ptr to string in
0477 022E C0C1          MOV    R1,R3            Set up addr. Reg. for GETV1
0478 0230 1602          JNE    STARY4           Not null ptr : o.k.
0479 0232 04D6          CLR    *R6              Clear the length bytes
0480 0234 1005          JMP    STARY6           Go on
0481 0236 0603  STARY4  DEC    R3               Point to string length
0482 0238 06A0          BL     @GETV1           Get the length from VDP
     023A 036C'
0483 023C 0981          SRL    R1,8             Make it two bytes
0484 023E C581          MOV    R1,*R6           Put the length in FAC area
0485            ******  FAC ENTRY HAS BEEN SET UP NOW  ********************
0486 0240       STARY6
0487            *
0488 0240 06A0          BL     @VGASSN          Do the VPUSH, GETSTR AND ASSIGN
     0242 0246'
0489            *****   ASSIGNMENT WILL POP THE STACK, OLD STACK ENTRY ******
0490            *****   DOES NOT NEED TO BE UPDATED IN THIS CASE     ********
0491 0244 0380          RTWP                    Return...
0492            ************************************************************
0493            ************************************************************
0494            ************************************************************
0495 0246 C08B  VGASSN  MOV    R11,R2           Save rtn addr.
0496            ******  VPUSH  routine in console ROM  ***********************
0497            ************************************************************
0498 0248 02E0          LWPI   GPLWS            Load GPL workspace.
     024A 83E0
```

```
0499 024C 020B          LI     R11,>1EAA      XML     table addr. For VPUSH
     024E 1EAA
0500 0250 069B          BL     *R11           Do the VPUSH
0501 0252 02E0          LWPI   UTILWS         Reload utility workspace.
     0254 0014'
0502             ************************************************************
0503             *****  GETS THE STRING SPACE, COPIES THE STRING FROM ERAM  **
0504             *****  INTO VDP STRING SPACE, AND SETS UP THE FAC WITH A   **
0505             *****  STRING ENTRY OF THE FOLLOWING FORM:                 **
0506             *****  ---------------------------------------------       **
0507             *****  | >001C | >65 | XX |  POINTER  |  LENGTH   |        **
0508             *****  |       |     |    | TO STRING | OF STRING |        **
0509             *****  ---------------------------------------------       **
0510             ***** FAC     +2    +3   +4          +6                    **
0511             ************************************************************
0512 0256 D199          MOVB   *R9,R6         R9:Addr. Of string (length byte
0513             *                             & string) supplies by ALC
0514 0258 0986          SRL    R6,8           Set up the length byte
0515 025A D806          MOVB   R6,@STATUS     MAKE SURE NO ACCIDENTAL ERROR EX
     025C 837C
0516 025E
0517 025E C806          MOV    R6,@BYTE       For GETSTR routine
     0260 830C
0518 0262 C806          MOV    R6,@FAC6       Set up length byte entry in FAC
     0264 8350
0519             ************  GETSTR routine in console GROM ****************
0520             ************************************************************
0521 0266 0420          BLWP   @GPLLNK        Call GETSTRG routine
     0268 0000
0522 026A 0038          DATA   >0038
0523             **************    SET UP FAC ENTRY         *****************
0524 026C 0206          LI     R6,FAC         Optimize to save bytes
     026E 834A
0525 0270 0204          LI     R4,>001C       Get addr. Of SREF
     0272 001C
0526 0274 CD84          MOV    R4,*R6+        Indicate a temporary string
0527 0276 DDA0          MOVB   @CBH65,*R6+    Indicates a string
     0278 0034'
0528 027A DD84          MOVB   R4,*R6+        Byte is not used
0529 027C C5A0          MOV    @SREF,*R6      Save ptr to the string
     027E 831C
0530             **************   COPY THE STRING TO STRING SPACE  **********
0531 0280 C220          MOV    @BYTE,R8       Get # of bytes to copy
     0282 830C
0532 0284 1309          JEQ    STRO4          If none to copy
0533 0286 C116          MOV    *R6,R4         Get pointer to destination
0534             *                             R4:addr.reg. for PUTV1 below
0535 0288 C0C9          MOV    R9,R3          r9:Ptr to string (length byte)
0536 028A 0583          INC    R3             Skip the len byte-pr to string
0537 028C D073  STRO6   MOVB   *R3+,R1        Get one byte from ERAM
0538 028E 06A0          BL     @PUTV1         Put 1 byte in VDP
     0290 0380'
0539 0292 0584          INC    R4             Update the destination addr.
0540 0294 0608          DEC    R8             1 less to move
0541 0296 15FA          JGT    STRO6          If not done-loop for more
0542             *                             ASSGNV destroys all the reg.
0543             *******     Call ASSNV routine in console ROM  ************
0544             ********
0545 0298 02E0  STRO4   LWPI   GPLWS          Load GPL workspace
     029A 83E0
```

```
0546 029C 020B          LI     R11,>1788      Access to ASSGN directly...
     029E 1788
0547 02A0 069B          BL     *R11           Call ASSGNV
0548 02A2 02E0          LWPI   UTILWS         Re-load utility workspace
     02A4 0014'
0549            ******  MOV    R2,R11         Restore RTN addr.
0550            ******  RT
0551 02A6 0452          B      *R2
```

```
                *
                *
                *       STRING REFERENCE UTILITY
                *
                *
                *         UTILITY ROUTINE SRENTR
                *          A utility to allow a ref to a string expression,
                *          a string variable or a string array element passed
                *          in a parameter list of LINK call.
                *
                *          The stack has info. For each argument.
                *          For a string expression, the stack entry looks
                *          like this
                *               ----------------------------------------
                *               | >001C     |>65|  |Pointer  | String  |
                *               |or Ptr to  |   |  |to string| Length  |
                *               |Value space|   |  |         |         |
                *               ----------------------------------------
                *               >001C : For a temporary string
                *               Ptr to Value Space : for a permanent string
                *          For a string variable, the stack entry looks like
                *          this ----------------------------------------
                *               |Ptr to     |>65|  |Pointer  | String  |
                *               |Value space|   |  |to string| Length  |
                *               ----------------------------------------
                *          NOTE: in this case, the pointer to string may
                *          no longer be valid.
                *          For a string array, the stack entry looks like
                *          this
                *               ----------------------------------------
                *               |Ptr to     |   |  |Ptr to   |         |
                *               |S.T. entry |   |  |Dim info.|         |
                *               ----------------------------------------
                *
                *          USER'S REGISTER USAGE -- the following registers must
                *           be set up prior to the call.
                *           R0 : Array element or zero
                *           R1 : Argument number
                *           R2 : String address in ERAM. (the first byte
                *                of the string is the length of the string.)
                *                Before calling SRENTR : The string len has
                *                the max. length which can be allocated for
                *                the string.
                *                After SRENTR call: The string len is the actual
                *                length of the string which is followed by the
                *                string itself.
                *
                ************************************************************
02A8 C01D       SRENTR  MOV    *R13,R0        For the change of the work space
                *                             Array element now is in R0
02AA C06D               MOV    @2(R13),R1     For the change of the work space
     02AC 0002
                *                             Argument no. is in R1
02AE 06A0               BL     @CHARG         Check valid argument no. and
     02B0 007E'
                *                             set up ptr to stack entry in R5
                * R3 points to parm type info address
02B2 D0D3               MOVB   *R3,R3         Get one byte identifier in CPU
02B4 0983               SRL    R3,8           Make it double
02B6 0603               DEC    R3             Flag 1 is for string expression
```

```
02B8 1302          JEQ    SRF01          String expression is o.k.
02BA 0643          DECT   R3             Flag 3 is for string variable
02BC 162C          JNE    STARYF         Not string exp. Or var., must be
                *                        string array in this case.
                *************  REF. TO STRING EXP. OR STRING VAR.  *********
02BE C000  SRF01   MOV    R0,R0          R0 must be 0 in this case
02C0 1631          JNE    SRFER2         ERROR : BAD ARGUMENT
02C2 C02D          MOV    @4(R13),R0     Now move string ptr into R0
     02C4 0004
02C6 C0C5          MOV    R5,R3          Addr.reg. for GETV1
02C8 05C3          INCT   R3             Point to the ID byte of the
                *                        stack entry
02CA 06A0          BL     @GETV1         Get 1 byte (ID) from VDP
     02CC 036C'
02CE 9801          CB     R1,@CBH65      Is this a string variable ?
     02D0 0034'
02D2 1626          JNE    SRFER1         No - STRING NO. MISMATCH ERROR
02D4 0643          DECT   R3             back to first word of entry
02D6 06A0          BL     @GET1
     02D8 0330'
02DA 0281          CI     R1,SREF-PAD    is this a temporary?
     02DC 001C
02DE 1302          JEQ    SRF02          if so, pointer on stack is OK.
02E0 C0C1          MOV    R1,R3          otherwise go back to symbol table
02E2 1002          JMP    SRF025
02E4 0223  SRF02   AI     R3,4           Pointe to "ptr to string"
     02E6 0004
02E8 06AD  SRF025  BL     @GET1          Get the "ptr to string" from
     02EA 0330'
                *                        stack entry in VDP
02EC C041  SRF03   MOV    R1,R1          Is it a null ptr?
02EE 1307          JEQ    SRF05          Yes - pot 0 in length byte
02F0 C181          MOV    R1,R6          R6 now contains "ptr to string"
02F2 0601          DEC    R1             Points to the length byte of
                *                        this string in string space
02F4 C0C1          MOV    R1,R3          Set up addr. reg. for GETV1
02F6 06A0          BL     @GETV1         Get the 1 byte actual length
     02F8 026C'
                *                        of this string in R1
02FA 9050          CB     *R0,R1         Compare the max. len. user
                *                        provided with the actual len.
02FC 1A15          JL     SRFER3         Not enough space - ERROR
                *                        STRING TRUNCATED ERROR
02FE DC01  SRF05   MOVB   R1,*R0+        Enough space - o.k.
                *                        Put the actual len. Into the
                *                        length byte
0300 1309          JEQ    SRF08          0 byte length - don't move
                ****** R0 : Points to the allocated string space in ERAM ***
                *                        Destination addr. for the
                *                        following move
0302 C0C6          MOV    R6,R3          Set up addr. reg. for GETV1
                *                        R3 now contains "ptr to string"
                *                        Source addr. for following move
0304 0981          SRL    R1,8           Make actual len. Byte a double
0306 C141          MOV    R1,R5          R5 : # of bytes to move
0308 06A0  SRF07   BL     @GETV1         Get 1 byte from VDP
     030A 036C'
030C DC01          MOVB   R1,*R0+        Put 1 byte into ERAM
030E 0583          INC    R3             Update the source addr.
0310 0605          DEC    R5             Update the counter
```

```
0662 0312 15FA          JGT    SRF07          More to move - loop back
0663 0314 0380  SRF08   RTWP                  Return the caller
0664            ************************************************************
0665            ***********   REF. TO STRING ARRAY       ********************
0666 0316 06A0  STARYF  BL     @STARYC        Check array element is within
     0318 01F4
0667            *       bound and find the string element ptr in value space
0668 031A C02D          MOV    @4(R13),R0     Put provided string addr.in ERAM
     031C 0004
0669            *  by the user into R0 now.  (for the change of work space)
0670 031E 10E6          JMP    SRF03          Go to the general code to move
0671            *                             the string from VDP to ERAM
0672            ************************************************************
0673 0320 0460  SRFER1  B      @NUERR1        STRING-NO. MISMATCH
     0322 010C
0674 0324 0460  SRFER2  B      @NUERR2        BAD ARGUMENT
     0326 0114
0675 0328 0200  SRFER3  LI     R0,ERRST       STRING TRUNCATED
     032A 2500
0676 032C 0460          B      @ERROR         Error out
     032E 039E
```

```
                  ************************************************************
                  ***      UTILITY ROUTINES TO ACCESS VDP                  ***
                  ************************************************************
                  ****** GET1 : Get two bytes of data from VDP
                  ******      R3 : Address in VDP
                  ******      R1 : Where the two bytes data stored
0330 06C3         GET1    SWPB   R3
0332 D803                 MOVB   R3,@VDPWA
     8C02
0336 06C3                 SWPB   R3
0338 D803                 MOVB   R3,@VDPWA
     8C02
033C 1000                 NOP
033E D060                 MOVB   @VDPRD,R1
     8800
0342 06C1                 SWPB   R1
0344 D060                 MOVB   @VDPRD,R1
     8800
0348 06C1                 SWPB   R1
034A 045B                 RT
                  ******  PUT1 : Put two bytes of data into VDP
                  ******     R4 : Address on VDP
                  ******     R1 : Data
034C 06C4         PUT1    SWPB   R4
034E D804                 MOVB   R4,@VDPWA
     8C02
0352 06C4                 SWPB   R4
0354 0264                 ORI    R4,WRVDP
     4000
0358 D804                 MOVB   R4,@VDPWA
     8C02
035C 1000                 NOP
035E D801                 MOVB   R1,@VDPWD
     8C00
0362 06C1                 SWPB   R1
0364 D801                 MOVB   R1,@VDPWD
     8C00
0368 06C1                 SWPB   R1
036A 045B                 RT
                  ******  GETV1 : Get 1 byte of data from VDP
                  ******       R3 : Address in VDP
                  ******       R1 : Where the 1 byte data stored
036C 06C3         GETV1   SWPB   R3
036E D803                 MOVB   R3,@VDPWA
     8C02
0372 06C3                 SWPB   R3
0374 D803                 MOVB   R3,@VDPWA
     8C02
0378 1000                 NOP
037A D060                 MOVB   @VDPRD,R1
     8800
037E 045B                 RT
                  ******  PUTV1 : Put 1 bytes of data into VDP
                  ******       R4 : Address on VDP
                  ******       R1 : Data
0380 06C4         PUTV1   SWPB   R4
0382 D804                 MOVB   R4,@VDPWA
     8C02
0386 06C4                 SWPB   R4
0388 0264                 ORI    R4,WRVDP
```

```
     038A 4000
 0725 038C D804          MOVB   R4,@VDPWA
      038E 8C02
 0726 0390 1000          NOP
 0727 0392 D801          MOVB   R1,@VDPWD
      0394 8C00
 0728 0396 045B          RT
```

```
 0730            ********            ********
 0731            **     ERROR ROUTINE      **
 0732            ********            ********
 0733 039B C81D  ERENTR MOV    *R13,@ERRCOD
      039A 8322
 0734 039C 1002         JMP    ERR2
 0735 039E C800  ERROR  MOV    R0,@ERRCOD
      03A0 8322
 0736 03A2 02E0  ERR2   LWPI   GPLWS
      03A4 83E0
 0737 03A6 020B         LI     R11,UTLTAB
      03A8 0000
 0738 03AA 022B         AI     R11,14
      03AC 000E
 0739 03AE C2DB         MOV    *R11,R11       restore GPL return address
 0740 03B0 0460         B      @SET
      03B2 00CE
 0741                   END
NO ERRORS,     NO WARNINGS
```

```
ARY1       00C2'  0206  0198  0455
ARYBND     00A2'  0187  0162  0338
BASE       8343   0021  0213
BYTE       830C   0015  0430  0517  0531
CBH65      0034'  0086  0403  0474  0527  0623
CKARG      007E'  0169  0128  0289  0385  0605
COUNT      8312   0018  0172
CRULST     83D0   0033
EØBIT      0035'  0087
ERENTR     0398'  0733  0076
ERR     D  0010'  0076  0071
ERR2       03A2'  0736  0734
ERRBA      1600   0050  0250  0438
ERRBLN     1A00   0054
ERRBN      1100   0044
ERRBS      1700   0051  0252
ERRBV      1300   0047
ERRCC      1200   0046
ERRCDT     1900   0053
ERRCOD     8322   0020  0733  0735
ERRDAT     1F00   0059
ERRFE      1D00   0057
ERRFNE     1B00   0055
ERRINP     1E00   0058
ERRIOB     1C00   0056
ERRIS      1000   0043
ERRLTL     2000   0060
ERRMM      2100   0045
ERRNC      1800   0052
ERRNO      2300   0062
ERRNTB     1400   0048
ERROR      039E'  0735  0249  0251  0253  0437  0439  0676
ERRSNF     0F00   0064
ERRSNM     1500   0049  0248  0436
ERRST      2500   0065  0675
SRRSYN     2200   0061
ERRUC      2400   0063
EXBASX     6010   0012
FAC        834A   0023  0027  0152  0303  0327  0412  0472  0524
FAC6       8350   0027  0518
FLAG       8348   0022
GET1       0330'  0684  0149  0192  0223  0228  0305  0324  0329  0415  0447
                  0463  0626  0632
GETV1      036C'  0711  0145  0195  0320  0402  0450  0482  0622  0640  0658
GPLLNK  R  0268'  0070  0521
GPLWS      83E0   0034  0498  0545  0736
MAXVDP     8370   0029
NAENTR     0036'  0126  0072
NRENTR     0124'  0287  0073
NUERR1     010C'  0248  0147  0189  0196  0343  0673
NUERR2     0114'  0250  0134  0141  0170  0173  0344  0674
NUERR3     011C'  0252  0217  0234  0243
NUM00      0060'  0151  0164
NUM01      0066'  0153
NUM02      006A'  0154  0158
NUM04      0078'  0162  0138
NUM08      00D8'  0220  0215
NUM10      00EA'  0226  0238
NUMASG  D  0000'  0072  0071
NUMREF  D  0004'  0073  0071
```

```
OLDS      8310   0016  0179
PAD       8300   0014  0627
PARM      200A   0013  0183
PUT1      034C'  0697  0155  0432
PUTV1     0380'  0721  0538
R0        0000         0126  0140  0140  0200  0202  0203  0216  0219  0242
                       0248  0250  0252  0287  0300  0300  0315  0315  0381
                       0397  0397  0436  0438  0459  0461  0462  0601  0616
                       0616  0618  0642  0646  0659  0668  0675  0735
R1        0001         0127  0150  0154  0169  0169  0171  0172  0174  0175
                       0181  0193  0207  0210  0212  0224  0229  0230  0231
                       0288  0306  0325  0330  0382  0403  0416  0430  0448
                       0476  0477  0483  0484  0537  0603  0623  0627  0629
                       0634  0634  0636  0637  0639  0642  0646  0656  0657
                       0659  0689  0690  0691  0692  0703  0704  0705  0706
                       0716  0727
R10       000A         0206  0246
R11       000B         0187  0206  0443  0495  0499  0500  0546  0547  0737
                       0738  0739  0739
R13       000D         0126  0127  0287  0288  0381  0382  0383  0601  0603
                       0618  0668  0733
R2        0002         0153  0157  0302  0308  0328  0332  0443  0464  0495
                       0551
R2LB      0019'  0081
R3        0003         0132  0132  0136  0137  0142  0143  0148  0150  0151
                       0152  0154  0181  0182  0183  0188  0191  0193  0203
                       0221  0222  0224  0225  0226  0244  0294  0294  0296
                       0304  0307  0312  0317  0318  0323  0323  0331  0388
                       0388  0390  0391  0394  0399  0400  0413  0417  0444
                       0446  0448  0462  0473  0477  0481  0535  0536  0537
                       0608  0608  0609  0610  0612  0619  0620  0625  0629
                       0631  0639  0653  0660  0684  0685  0686  0687  0711
                       0712  0713  0714
R4        0004         0151  0156  0200  0213  0214  0230  0241  0303  0306
                       0327  0330  0412  0416  0428  0429  0459  0475  0525
                       0526  0528  0533  0539  0697  0698  0699  0700  0701
                       0721  0722  0723  0724  0725
R4LB      001D'  0082
R5        0005         0142  0175  0176  0177  0179  0191  0221  0304  0317
                       0399  0413  0428  0446  0619  0657  0661
R6        0006         0220  0231  0233  0233  0233  0239  0241  0242  0411
                       0418  0472  0473  0474  0475  0476  0479  0484  0512
                       0514  0515  0517  0518  0524  0526  0527  0528  0529
                       0533  0636  0653
R7        0007         0235
R8        0008         0212  0236  0531  0540
R9        0009         0187  0204  0383  0512  0535
REF00     0150'  0312  0297
REF01     016A'  0327  0340
REF02     0172'  0329  0333
REF03     0142'  0305  0309
REF04     0180'  0338  0313
RFERR1    0186'  0343  0322
RFERR2    018A'  0344  0301  0316
SAENTR    018E'  0381  0074
SCLEN     8355   0025
SCNAME    8356   0026
SCNKEY    000E   0011
SCTEMP    8354   0024
SET       00CE   0007  0740
```

```
SNDADD     83CC    0031
SREF       831C    0019  0529  0627
SRENTR     02A8'   0601  0075
SRF01      02BE'   0616  0611
SRF02      02E4'   0631  0628
SRF025     02E8'   0632  0630
SRF03      02EC'   0634  0670
SRF05      02FE'   0646  0635
SRF07      0308'   0658  0662
SRF08      0314'   0663  0649
SRFER1     0320'   0673  0624
SRFER2     0324'   0674  0617
SRFER3     0328'   0675  0644
STARY      021C'   0467  0395
STARY1     020C'   0453  0451
STARY4     0236'   0481  0478
STARY6     0240'   0486  0480
STARYC     01F4'   9443  0467  0666
STARYF     0316'   0666  0613
STATUS     837C    0030  0515
STERR1     01E4'   0436  0404  0445  0452
STERR2     01EC'   0438  0392  0398
STFLGS     83CE    0032
STR02      01C4'   0415  0419
STR04      0298'   0545  0532
STR06      028C'   0537  0541
STRASG  D  0008'   0074  0071
STRREF  D  000C'   0075  0071
STRTN      01E2'   0433
UTILWS     0014'   0080  0072  0073  0074  0075  0076  0081  0082  0501  0548
UTLTAB  R  03A8'   0070  0737
VDPRD      8800    0037  0689  0691  0716
VDPWA      8C02    0039  0685  0687  0698  0701  0712  0714  0722  0725
VDPWD      8C00    0038  0703  0705  0727
VGASSN     0246'   0495  0422  0488
VSPTR      836E    0028
WRVDP      4000    0036  0700  0724
```

```
SOURCE ACCESS NAME=     SD2.ASSM.SRC.SAVE2
OBJECT ACCESS NAME=     SD2.ASSM.OBJ.SAVE2
LISTING ACCESS NAME=    SD2.ASSM.LST.SAVE2
ERROR ACCESS NAME=
OPTIONS=                XREF,BUNLST,TUNLST,DUNLST
MACRO LIBRARY PATHNAME=
```

```
0001                    IDT    'SAVE'
0002             *-------------------------------------------------------
0003             *
0004             *      Save memory image to cassette or disk
0005             *
0006             *      The user must load his own program in to CPU
0007             *      RAM, and then load this utility.  He must put
0008             *      a DEF START,STOP in his program.  This utility will
0009             *      prompt for a file name.
0010             *
0011             *-------------------------------------------------------
0012                    REF    VSBW,VSBR,VMBW,GPLLNK
0013                    REF    DSRLNK,KSCAN
0014                    REF    GRMWA,GRMRA,GRMRD,GPLWS
0015                    DEF    SAVE
0016                    REF    SFIRST,SLAST,SLOAD,VMBR
0017 2800               AORG   >2800
0018             *
0019 2800   28          TEXT   '(C) COPYRIGHT 1981 BY TEXAS INSTRUMENTS, INC.'
0020 282E               EVEN
0021             *
0022       8356  SCNAME EQU    >8356
0023       837C  STATUS EQU    >837C
0024       8375  KEYCOD EQU    >8375
0025             *
0026 282E   20   MENU   TEXT   '                                '            1
0027 284E   20          TEXT   '  SAVE MEMORY IMAGE             '            2
0028 286E   20   BLANK  TEXT   '                                '            3
0029 288E   20          TEXT   ' 1. PROGRAM TO BE SAVED MUST    '            4
0030 28AE   20          TEXT   '    ALREADY BE LOADED.          '            5
0031 28CE   20          TEXT   '                                '            6
0032 28EE   20          TEXT   ' 2. THE FIRST WORD IN THE       '            7
0033 290E   20          TEXT   '    PROGRAM MUST BE EXECUTABLE. '            8
0034 292E   20          TEXT   '                                '            9
0035 294E   20          TEXT   ' 3. PROGRAMS WHICH ARE BIGGER   '           10
0036 296E   20          TEXT   '    THAN 8K WILL CREATE NEW     '           11
0037 298E   20          TEXT   '    FILE NAMES BY ADDING A ONE  '           12
0038 29AE   20          TEXT   '    TO THE LAST BYTE OF THE     '           13
0039 29CE   20          TEXT   '    CURRENT FILE NAME.          '           14
0040 29EE   20          TEXT   '                                '           15
0041 2A0E   20          TEXT   ' CURRENT FILE :                 '           16
0042       0202  SADDR  EQU    $-MENU+2
0043 2A2E   20          TEXT   ' STATUS:                        '           17
0044 2A4E   20          TEXT   '                                '           18
0045       0242  EADDR  EQU    $-MENU+2
0046 2A6E   20          TEXT   '                                '           19
0047 2A8E   20          TEXT   '                                '           20
0048 2AAE   20          TEXT   '                                '           21
0049 2ACE   20          TEXT   '                                '           22
0050 2AEE   20          TEXT   '  ENTER FILE NAME?              '           23
0051 2B0E   20          TEXT   '                                '           24
0052       0300  MENLEN EQU    $-MENU
0053       02C2  PADDR  EQU    22*32+2
0054       02E2  FADDR  EQU    (23*32)+2
0055 2B2E  4353  CS     DATA   'CS'
0056 2B30   01   ONE    BYTE   1
0057 2B31   00          BYTE   0
0058 2B32               EVEN
0059       2B31  FILEN  EQU    $-1
0060 2B32        FILE   BSS    32              MUST BE ON EVEN WORD BOUNDARY
```

```
0061      2B52  FILEND  EQU    $
0062            *RR1    TEXT   ' SFIRST IS LESS THAN >A000      '
0063 2B52   20  ERR2    TEXT   ' SLAST IS INVALID               '
0064 2B72   20  ERR3    TEXT   ' I/O ERROR                      '
0065 2B92   20  ERR4    TEXT   ' BAD FILE NAME                  '
0066 2BB2 0000  ERRCOD  DATA   0
0067 2BB4 0600  CPAB    DATA   >0600,>1000,0
0068 2BBA 2000  FSIZE   DATA   >2000,0
0069      0F80  PAB     EQU    >F80
0070      0F89  PABFL   EQU    >F80+9
0071      0F8A  PABFN   EQU    >F80+10
0072 2BBE 0000  HEADER  DATA   0,0,0
0073      0300  SAB     EQU    >300
0074 2BC4               EVEN
0075 2BC4   B7  INITS   BYTE   >B7,>F,>80,>F,>D0
0076      2BC5  POS     EQU    INITS+1
0077      2BC8  OFFS    EQU    $-1
0078 2BCA               EVEN
0079 2BCA   B7  ON      BYTE   >B7,>F
0080 2BCC   00  CURSOR  BYTE   0,0,0,0,0,0,>7C,>7C
0081 2BD4 0000  LOAD    DATA   0
```

```
0083                *-------------------------------------------------------------
0084                *
0085                *      SCANS KEY BOARD FOR FILE NAME INPUT
0086                *
0087                *-------------------------------------------------------------
0088 2BD6           SAVE
0089 2BD6 04C0              CLR    R0
0090 2BD8 0201              LI     R1,MENU
     2BDA 282E
0091 2BDC 0202              LI     R2,MENLEN
     2BDE 0300
0092 2BE0 0420              BLWP   @VMBW
     2BE2 0000
0093 2BE4 0200      SAVE05  LI     R0,>400            SET UP SPRITE
     2BE6 0400
0094 2BE8 0201              LI     R1,CURSOR
     2BEA 2BCC
0095 2BEC 0202              LI     R2,8
     2BEE 0008
0096 2BF0 0420              BLWP   @VMBW
     2BF2 2BE2
0097 2BF4 C820              MOV    @ON,@INITS         TURN SPRITES ON
     2BF6 2BCA
     2BF8 2BC4
0098                *
0099 2BFA           SAV20
0100 2BFA 0203              LI     R3,FADDR           R3 -> CURRENT CHAR POSITION
     2BFC 02E2
0101 2BFE C820              MOV    @ON,@INITS         INITIALIZE SPRITES
     2C00 2BCA
     2C02 2BC4
0102 2C04           SAV30
0103 2C04 06A0              BL     @WSPRIT            WRITE SPRITE
     2C06 2C6C
0104 2C08 0420              BLWP   @KSCAN             WAIT FOR A KEY
     2C0A 0000
0105 2C0C D0A0              MOVB   @STATUS,R2         IS THIS A NEW KEY?
     2C0E 837C
0106 2C10 1603              JNE    SAV40              YES
0107 2C12 10F8              JMP    SAV30              NO. WAIT
0108 2C14 0460      EXIT    B      @DONE
     2C16 2DF8
0109                *
0110 2C18           SAV40
0111 2C18 04C1              CLR    R1
0112 2C1A D060              MOVB   @KEYCOD,R1         R1 -> KEY
     2C1C 8375
0113 2C1E 0281              CI     R1,>0800           NO, BACK SPACE?
     2C20 0800
0114 2C22 1317              JEQ    BACKSP
0115 2C24 0281              CI     R1,>0D00           ENTER KEY ?
     2C26 0D00
0116 2C28 132A              JEQ    ENTER
0117 2C2A 0281              CI     R1,>0500
     2C2C 0500
0118 2C2E 13F2              JEQ    EXIT
0119 2C30 0281              CI     R1,>0F00
     2C32 0F00
0120 2C34 13EF              JEQ    EXIT
0121 2C36 0281              CI     R1,>2000
```

```
     2C38 2000
0122 2C3A 1AE4          JL     SAV30
0123 2C3C C003          MOV    R3,R0              WRITE KEY ON SCREEN
0124 2C3E 0420          BLWP   @VSSW
     2C40 0000
0125 2C42 0583          INC    R3                 BUMP CURSOR ADDRESS
0126 2C44 0283          CI     R3,24*32-1
     2C46 02FF
0127 2C48 1BD8          JH     SAV20
0128 2C4A B820          AB     @EIGHT,@POS        MOVE SPRITE OVER
     2C4C 2C5C
     2C4E 2BC5
0129 2C50 10D9          JMP    SAV30              NO, GET NEXT KEY
0130 2C52       BACKSP
0131 2C52 0283          CI     R3,FADDR           AT THE BEGINNING?
     2C54 02E2
0132 2C56 12D6          JLE    SAV30              YES, IGNORE KEY
0133 2C58 0603          DEC    R3                 NO, BACK UP
0134 2C5A 0201          LI     R1,>0800
     2C5C 0800
0135      2C5C  EIGHT   EQU    $-2
0136 2C5E 7801          SB     R1,@POS            ADJUST SPRITE
     2C60 2BC5
0137 2C62 10D0          JMP    SAV30              GET NEXT KEY
0138 2C64 0201  ENAME   LI     R1,ERR4
     2C66 2B92
0139 2C68 0460          B      @EI010
     2C6A 2D9A
0140 2C6C       WSPRIT
0141 2C6C 0200          LI     R0,SAB
     2C6E 0300
0142 2C70 0201          LI     R1,INITS
     2C72 2BC4
0143 2C74 0202          LI     R2,5
     2C76 0005
0144 2C78 0420          BLWP   @VMBW
     2C7A 2BF2
0145 2C7C 045B          RT
0146 2C7E       ENTER
0147 2C7E D820          MOVB   @OFFS,@INITS
     2C80 2BC8
     2C82 2BC4
0148 2C84 06A0          BL     @WSPRIT
     2C86 2C6C
0149 2C88 0200          LI     R0,FADDR
     2C8A 02E2
0150 2C8C 0201          LI     R1,FILE
     2C8E 2B32
0151 2C90 0202          LI     R2,32
     2C92 0020
0152 2C94 0420          BLWP   @VMBR              HEAD FILE NAME
     2C96 0000
```

```
0154                *-------------------------------------------------------------
0155                *
0156                *      SCAN FILE NAME FOR LENGTH
0157                *
0158                *      SET UP R6 -> CURRENT START ADDRESS
0159                *             R7 -> CURRENT #BYTES LEFT TO SAVE
0160                *             R8 -> LAST BYTE IN CPU FILE MANE BUFFER
0161                *             R4 -> LENGTH OF THE FILE NAME + 1
0162                *
0163                *-------------------------------------------------------------
0164 2C98 0203          LI     R3,FILE
     2C9A 2B32
0165 2C9C 0283  SAV45   CI     R3,FILEND         SCAN FOR END OF FILE NAME.
     2C9E 2B52
0166 2CA0 15E1          JGT    ENAME
0167 2CA2 9833          CB     *R3+,@BLANK
     2CA4 286E
0168 2CA6 16FA          JNE    SAV45
0169 2CA8 C203          MOV    R3,R8             R8 -> LAST BYTE IN FILE NAME
0170 2CAA 0648          DECT   R8
0171 2CAC 0204          LI     R4,FILE
     2CAE 2B32
0172 2CB0 60C4          S      R4,R3             CALCULATE LENGTH OF FILE NAME
0173 2CB2 C103          MOV    R3,R4             R4 -> # BYTES TO MOVE INTO VDP RAM
0174 2CB4 0603          DEC    R3                R3 = LENGTH BYTE
0175 2CB6 0283          CI     R3,3
     2CB8 0003
0176 2CBA 11D4          JLT    ENAME             BAD DEVICE NAME
0177 2CBC 06C3          SWPB   R3
0178 2CBE D803          MOVB   R3,@FILEN
     2CC0 2B31
0179                *
0180 2CC2 0206          LI     R6,SLOAD          PROGRAM'S LOAD ADDRESS
     2CC4 0000
0181 2CC6 C806          MOV    R6,@LOAD
     2CC8 2BD4
0182 2CCA 0206          LI     R6,SFIRST         GET PROGRAM'S START ADDRESS
     2CCC 0000
0183                *      CI     R6,>A000          VALID?
0184                *      JL     ESTART            NO
0185 2CCE 0207          LI     R7,SLAST          GET PROGRAM'S STOP ADDRESS
     2CD0 0000
0186 2CD2 61C6          S      R6,R7             FILE SIZE
0187 2CD4 115B          JLT    ESTOP             BAD FILE SIZE
```

```
0189                *-------------------------------------------------------------
0190                *
0191                *      MAIN LOOP TO SAVE EACH FILE IN MEMORY IMAGE FORMAT
0192                *           CALCULATES FILE SIZE
0193                *           BUILDS HEADER
0194                *           WRITES PAB
0195                *           CHECKS TO SEE IF NEED TO WRITE ANOTHER FILE
0196                *
0197                *-------------------------------------------------------------
0198 2CD6 C087      MAIN    MOV    R7,R2              #BYTES TO MOVE
0199 2CD8 0287              CI     R7,>2000-6         BIGGER THAN 8K?
     2CDA 1FFA
0200 2CDC 1102              JLT    MAIN10             NOPE
0201 2CDE 0202              LI     R2,>2000-6
     2CE0 1FFA
0202 2CE2 C282      MAIN10  MOV    R2,R10             R10 = # BYTES TO MOVE
0203 2CE4 0222              AI     R2,6               MAKE ROOM FOR HEADER
     2CE6 0006
0204 2CE8 C802              MOV    R2,@FSIZE
     2CEA 2BBA
0205 2CEC 0200              LI     R0,PAB
     2CEE 0F80
0206 2CF0 0201              LI     R1,CPAB
     2CF2 2BB4
0207 2CF4 0202              LI     R2,9
     2CF6 0009
0208 2CF8 0420              BLWP   @VMBW              WRITE 10 BYTES OF PAB
     2CFA 2C7A
0209                *
0210 2CFC C084              MOV    R4,R2              #BYTES IN FILE NAME
0211 2CFE 0200              LI     R0,PABFL
     2D00 0F89
0212 2D02 0201              LI     R1,FILEN           MOVE FILENAME INTO PAB
     2D04 2B31
0213 2D06 0420              BLWP   @VMBW
     2D08 2CFA
0214 2D0A 04C0              CLR    R0
0215 2D0C 0201              LI     R1,MENU
     2D0E 282E
0216 2D10 0202              LI     R2,MENLEN          RESTORE SCREEN
     2D12 0300
0217 2D14 0420              BLWP   @VMBW
     2D16 2D08
0218 2D18 0200              LI     R0,SADDR
     2D1A 0202
0219 2D1C 0201              LI     R1,FILE
     2D1E 2B32
0220 2D20 C084              MOV    R4,R2
0221 2D22 0602              DEC    R2
0222 2D24 0420              BLWP   @VMBW
     2D26 2D16
0223                *
0224 2D28 04E0              CLR    @HEADER
     2D2A 2BBE
0225 2D2C 81CA              C      R10,R7             MORE DATA TO WRITE?
0226 2D2E 1302              JEQ    MAIN20             NO
0227 2D30 0720              SETO   @HEADER            YES, SET FLAG TO INDICATE THIS
     2D32 2BBE
0228 2D34           MAIN20
0229 2D34 C820              MOV    @FSIZE,@HEADER+2   LOAD FILE SIZE
```

```
     2D36 2BBA
     2D38 2BC0
0230 2D3A C820          MOV    @LOAD,@HEADER+4   LOAD START ADDRESS
     2D3C 2BD4
     2D3E 2BC2
0231 2D40 0200          LI     R0,>1000
     2D42 1000
0232 2D44 0201          LI     R1,HEADER
     2D46 2BBE
0233 2D48 0202          LI     R2,6
     2D4A 0006
0234 2D4C 0420          BLWP   @VMBW             WRITE HEADER IN PABBUF
     2D4E 2D26
0235            *
0236 2D50 0200          LI     R0,>1006          WRITE AFTER THE HEADER
     2D52 1006
0237 2D54 C08A          MOV    R10,R2            R10=SIZE
0238 2D56 C046          MOV    R6,R1             R6=START
0239 2D58 0420          BLWP   @VMBW             MOVE PROGRAM INTO VDP
     2D5A 2D4E
0240            *
0241 2D5C 8820          C      @FILE,@CS         CASSETTE?
     2D5E 2B32
     2D60 2B2E
0242 2D62 1323          JEQ    CASSIO
0243 2D64       DISKIO
0244 2D64 0201          LI     R1,PABFL          DO A MEMORY IMAGE SAVE
     2D66 0F89
0245 2D68 C801          MOV    R1,@SCNAME
     2D6A 8356
0246 2D6C 0420          BLWP   @DSRLNK
     2D6E 0000
0247 2D70 0008          DATA   8
0248 2D72 130F          JEQ    EIO               DISK ERROR
0249 2D74 06A0          BL     @CKERR
     2D76 2E0A
0250 2D78 C060          MOV    @HEADER,R1
     2D7A 2BBE
0251 2D7C 133D          JEQ    DONE
0252 2D7E A18A          A      R10,R6            CALCULATE NEW START
0253 2D80 61CA          S      R10,R7            CALCULATE NEW LENGTH
0254 2D82 A80A          A      R10,@LOAD         CALCULATE NEW LOAD ADDRESS
     2D84 2BD4
0255 2D86 B620          AB     @ONE,*R8          CHANGE FILE NAME
     2D88 2B30
0256 2D8A 10A5          JMP    MAIN
0257            *
0258 2D8C 0201  ESTOP   LI     R1,ERR2           BAD STOP ADDRESS
     2D8E 2B52
0259 2D90 1004          JMP    EIO10
0260            *START  LI     R1,ERR1           BAD START ADDRESS
0261            *       JMP    EIO10
0262 2D92 06A0  EIO     BL     @CKERR
     2D94 2E0A
0263 2D96 0201  EIO05   LI     R1,ERR3           IO ERROR
     2D98 2B72
0264 2D9A 0200  EIO10   LI     R0,EADDR
     2D9C 0242
0265 2D9E 0202          LI     R2,32
     2DA0 0020
```

```
0266 2DA2 0420          BLWP   @VMBW
     2DA4 2D5A
0267 2DA6 0460          B      @SAVED5
     2DA8 2BE4
0268             *
0269 2DAA        CASSIO
0270 2DAA 0201          LI     R1,PAB+13          POINT TO END OF "CS1"
     2DAC 0F8D
0271 2DAE C801          MOV    R1,@SCNAME
     2DB0 8356
0272 2DB2 0201          LI     R1,3
     2DB4 0003
0273 2DB6 C801          MOV    R1,@>8354          NAME IS 3 BYTES LONG
     2DB8 8354
0274 2DBA 0201          LI     R1,>800            OFFSET FOR SEARCH
     2DBC 0800
0275 2DBE D801          MOVB   R1,@>836D
     2DC0 836D
0276 2DC2 0200          LI     R0,PABFN           MOVE DEVICE NAME TO FAC
     2DC4 0F8A
0277 2DC6 0201          LI     R1,>834A
     2DC8 834A
0278 2DCA 0202          LI     R2,3
     2DCC 0003
0279 2DCE 0420          BLWP   @VMBR
     2DD0 2C96
0280             *
0281 2DD2 04E0          CLR    @>83D0             CRULST, INITIALIZE SEARCH
     2DD4 83D0
0282 2DD6 D820          MOVB   @>8300,@>837C
     2DD8 83D0
     2DDA 837C
0283 2DDC 0420          BLWP   @GPLLNK
     2DDE 0000
0284 2DE0 003D          DATA   >3D
0285             *
0286 2DE2 06A0          BL     @CKERR
     2DE4 2E0A
0287 2DE6 C060          MOV    @HEADER,R1         FINISHED?
     2DE8 2BBE
0288 2DEA 1306          JEQ    DONE               YES
0289 2DEC A18A          A      R10,R6             UPDATE START
0290 2DEE A80A          A      R10,@LOAD
     2DF0 2BD4
0291 2DF2 61CA          S      R10,R7             UPDATE # BYTES LEFT
0292 2DF4 0460          B      @MAIN
     2DF6 2CD6
0293             *-------------------------------------------------------------
0294 2DF8        DONE
0295 2DF8 04C0          CLR    R0
0296 2DFA D800          MOVB   R0,@>837C          CLEAR STATUS
     2DFC 837C
0297 2DFE 02E0          LWPI   GPLWS
     2E00 0000
0298 2E02 0300          LIMI   2
     2E04 0002
0299 2E06 06A0          BL     @>70               RETURN TO GPL
     2E08 0070
0300             *
0301 2E0A        CKERR
```

```
 0302 2E0A 0200          LI     R0,PAB+1          STATUS BYTE
      2E0C 0F81
 0303 2E0E 0420          BLWP   @VSBR             READ 1 BYTE FROM VDP
      2E10 0000
 0304 2E12 09D1          SRL    R1,13             RIGHT JUSTIFY ERROR CODE
 0305 2E14 C041          MOV    R1,R1
 0306 2E16 1306          JEQ    CKRT              ERROR==0
 0307 2E18 06C1          SWPB   R1
 0308 2E1A 0221          AI     R1,>3000          CONVERT ERROR TO ASCII
      2E1C 3000
 0309 2E1E D801          MOVB   R1,@ERR3+11       PUT IN ERROR MESSAGE
      2E20 2B7D
 0310 2E22 10B9          JMP    EI005
 0311 2E24 045B  CKRT    RT
 0312                    END
NO ERRORS,     NO WARNINGS
```

$	2E26		0042	0045	0052	0059	0061	0077	0135	
BACKSP	2C52	0130	0114							
BLANK	286E	0028	0167							
CASSIO	2DAA	0269	0242							
CKERR	2E0A	0301	0249	0262	0286					
CKRT	2E24	0311	0306							
CPAB	2BB4	0067	0206							
CS	2B2E	0055	0241							
CURSOR	2BCC	0080	0094							
DISKIO	2D64	0243								
DONE	2DF8	0294	0108	0251	0288					
DSRLNK R	2D6E	0013	0246							
EADDR	0242	0045	0264							
EIGHT	2C5C	0135	0128							
EIO	2D92	0262	0248							
EIO05	2D96	0263	0310							
EIO10	2D9A	0264	0139	0259						
ENAME	2C64	0138	0166	0176						
ENTER	2C7E	0146	0116							
ERR2	2B52	0063	0258							
ERR3	2B72	0064	0263	0309						
ERR4	2B92	0065	0138							
ERRCOD	2BB2	0066								
ESTOP	2D8C	0258	0187							
EXIT	2C14	0108	0118	0120						
FADDR	02E2	0054	0100	0131	0149					
FILE	2B32	0060	0150	0164	0171	0219	0241			
FILEN	2B31	0059	0178	0212						
FILEND	2B52	0061	0165							
FSIZE	2BBA	0068	0204	0229						
GPLLNK R	2DDE	0012	0283							
GPLWS R	2E00	0014	0297							
GRMRA R		0014								
GRMRD R		0014								
GRMWA R		0214								
HEADER	2BBE	0072	0224	0227	0229	0230	0232	0250	0287	
INITS	2BC4	0075	0076	0097	0101	0142	0147			
KEYCOD	8375	0024	0112							
KSCAN R	2C0A	0013	0104							
LOAD	2BD4	0081	0181	0230	0254	0290				
MAIN	2CD6	0198	0256	0292						
MAIN10	2CE2	0202	0200							
MAIN20	2D34	0228	0226							
MENLEN	0300	0052	0091	0216						
MENU	282E	0026	0042	0045	0052	0090	0215			
OFFS	2BC8	0077	0147							
ON	2BCA	0079	0097	0101						
ONE	2B30	0056	0255							
PAB	0F80	0069	0205	0270	0302					
PABFL	0F89	0070	0211	0244						
PABFN	0F8A	0071	0276							
PADDR	02C2	0053								
POS	2BC5	0076	0128	0136						
R0	0000		0089	0093	0123	0141	0149	0205	0211	0214	0218
			0231	0236	0264	0276	0295	0296	0302		
R1	0001		0090	0094	0111	0112	0113	0115	0117	0119	0121
			0134	0136	0138	0142	0150	0206	0212	0215	0219
			0232	0238	0244	0245	0250	0258	0263	0270	0271
			0272	0273	0274	0275	0277	0287	0304	0305	0305
			0307	0308	0309						

```
R10        000A          0202  0225  0237  0252  0253  0254  0289  0290  0291
R2         0002          0091  0095  0105  0143  0151  0198  0201  0202  0203
                         0204  0207  0210  0216  0220  0221  0233  0237  0265
                         0278
R3         0003          0100  0123  0125  0126  0131  0133  0164  0165  0167
                         0169  0172  0173  0174  0175  0177  0178
R4         0004          0171  0172  0173  0210  0220
R6         0006          0180  0181  0182  0186  0238  0252  0289
R7         0007          0185  0186  0198  0199  0225  0253  0291
R8         0008          0169  0170  0255
SAB        0300    0073  0141
SADDR      0202    0042  0218
SAV20      2BFA    0099  0127
SAV30      2C04    0102  0107  0120  0129  0132  0137
SAV40      2C18    0110  0106
SAV45      2C9C    0165  0168
SAVE    D  2BD6    0088  0015
SAVE05     2BE4    0093  0267
SCNAME     8356    0022  0245  0271
SFIRST  R  2CCC    0016  0182
SLAST   R  2CD0    0016  0185
SLOAD   R  2CC4    0016  0180
STATUS     837C    0023  0105
VMBR    R  2DD0    0016  0152  0279
VMBW    R  2DA4    0012  0092  0096  0144  0208  0213  0217  0222  0234  0239
                         0266
VSBR    R  2E10    0012  0303
VSBW    R  2C40    0012  0124
WSPRIT     2C6C    0140  0103  0148
```